

# PicoVRAndroidSDK\_UE4

# 开发说明文档

版本: v\_1.1.6

## 目录

1	SDK 简介.....	5
2	支持设备.....	5
2.1	VR 一体机.....	5
3	开发环境要求 .....	6
4	SDK 使用快速入门 .....	8
4.1	创建项目 .....	8
4.2	设置项目 .....	8
4.3	导入 SDK 开发包.....	10
4.4	完成项目 .....	11
4.5	项目打包 .....	12
5	控制器使用指南 .....	15
5.1	Pico Goblin 控制器 .....	15
5.1.1	使用指南.....	15
5.1.2	输入说明.....	17
5.1.3	相关蓝图节点 .....	19
5.2	Pico G2 控制器 .....	20
5.3	Pico Neo 控制器 .....	23
5.3.1	说明 .....	23
5.3.2	使用指南.....	23
5.3.3	输入说明.....	25
5.3.4	相关蓝图节点 .....	28

- 6 接口函数..... 32
  - 6.1 通用函数库..... 32
  - 6.2 专用函数库..... 32
- 7 支付系统..... 36
  - 7.1 准备工作..... 37
    - 7.1.1 申请并填入 APPKEY, APPID、SCOPE、DEVELOPERID、APP SECRET  
37
    - 7.1.2 设置回调代理事件..... 41
    - 7.1.3 用户登录..... 41
  - 7.2 其他相关接口..... 42
    - 7.2.1 PicoPaymentLogout..... 42
    - 7.2.2 PicoPaymentGetUserInfo..... 42
    - 7.2.3 PicoPaymentPaywithCoin..... 44
    - 7.2.4 PicoPaymentPayWithPayCode..... 46
    - 7.2.5 PicoPaymentQueryOrder..... 47
  - 7.3 开发者服务端交互..... 48
- 8 其他说明..... 53
  - 8.1 Goblin HMD 按键..... 53
  - 8.2 G2 HMD 按键..... 54
  - 8.3 Pico Neo HMD 按键..... 55
  - 8.4 Pico Neo 安全区域..... 55
  - 8.5 开启 Pico Neo 6 Dof 功能..... 57

9	已知问题.....	58
10	FAQ.....	59

# 1 SDK 简介

本文档介绍在 Unreal 游戏开发引擎环境下, 使用 PicoVRAndroidSDK\_UE4 (以下简称 SDK) 制作运行在 Pico 一体机设备上的 VR 应用。SDK 主要提供: 双目立体渲染, 光学畸变校正, 传感器融合, 异步时间扭曲、单缓冲渲染, 多交互支持(头手 3DOF/6DOF), 多种外设支持, 电源及散热管理, 账号及支付管理等功能。

SDK 以引擎插件的形式提供, 插件通过实现 UE4 的 VR 抽象层桥接引擎与 Pico 虚拟现实硬件设备。

这就使得引擎中 Camera 组件的位置/姿态将跟随 Pico 虚拟现实头盔运动, 使得引擎的 MotionController 组件将跟随 Pico 运动控制器运动。

当然, 调用 UE4 的 VR 抽象层的一些函数可以控制 Pico 虚拟实现硬件, 例如调用 Input/Head Mounted Display/Reset Orientation and Position 即可实现头戴显示器姿态/位置重置的功能。此外, 对于一些 Pico VR 特有的功能, 我们也提供了蓝图接口, 对于其调用方法也附带了 Demo 供开发者参考。

## 2 支持设备

### 2.1 VR 一体机

品牌	产品
小鸟看看	Pico Goblin、Pico G2、Pico Neo

SDK 不支持普通手机, 仅支持上述一体机。

### 3 开发环境要求

软件名称	版本信息
Unreal Engine	4.18.3、4.19.2、4.20.3、4.21.2
Visual Studio	2015 及以上 (对于 UE4.20+, 请务必安装最新版 VS2017)
JDK	jdk1.7.0_01 及以上
Android Works	尽量使用最新版本

SDK 对于硬件的 ROM 版本没有特殊要求，但尽量确保为最新版。

Android Works 请参照：

<https://docs.unrealengine.com/en-US/Platforms/Mobile/Android/InstallingAndroidCodeWorksAndroid>

安装。

另外，在安装 Visual Studio 2015 时务必勾选 common tools for visual c++ 2015，否则无法编译项目且影响打包。

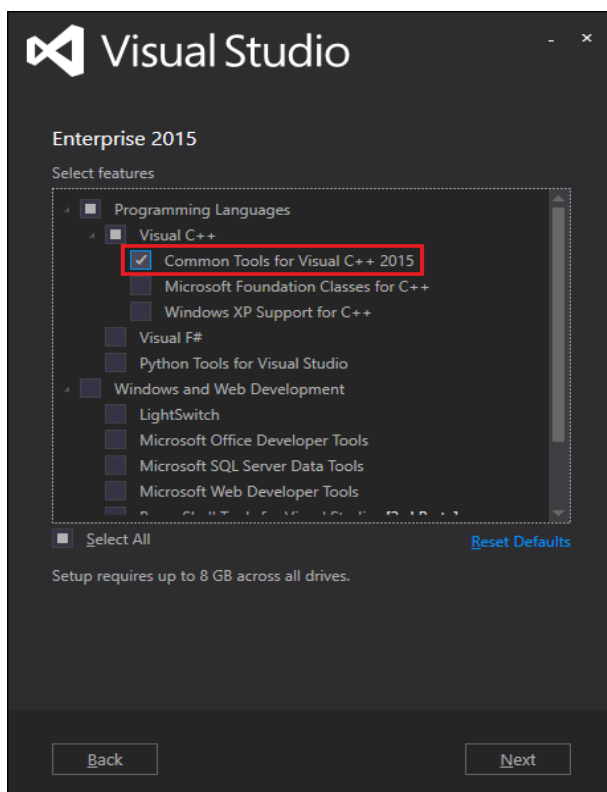


图 3.1 Visual Studio 2015 安装选项

在安装 Visual Studio 2017 时务必勾选 “Game development with C++” :

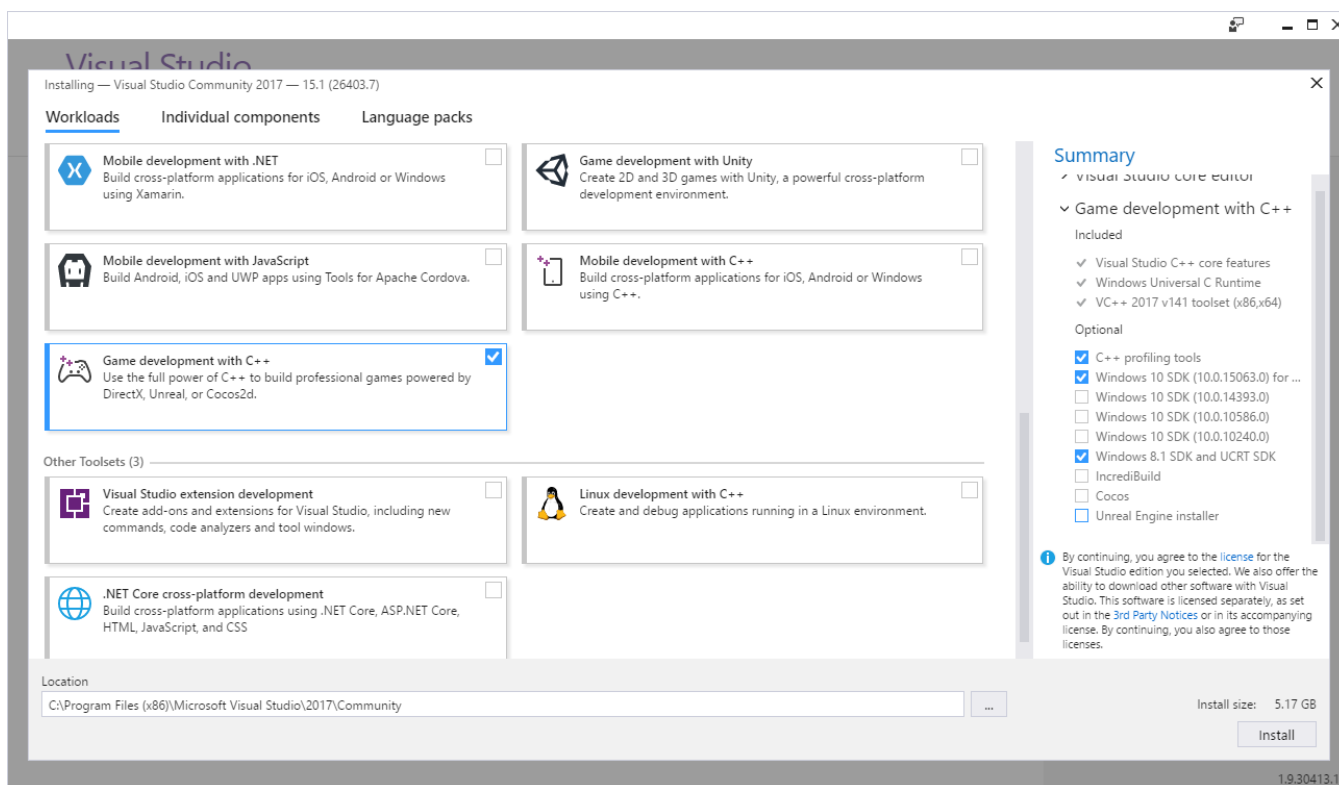


图 3.2 Visual Studio 2017 安装选项

## 4 SDK 使用快速入门

### 4.1 创建项目

新建项目时选择蓝图及 C++ 项目均可，这里以蓝图 Blank 项目模板为例：

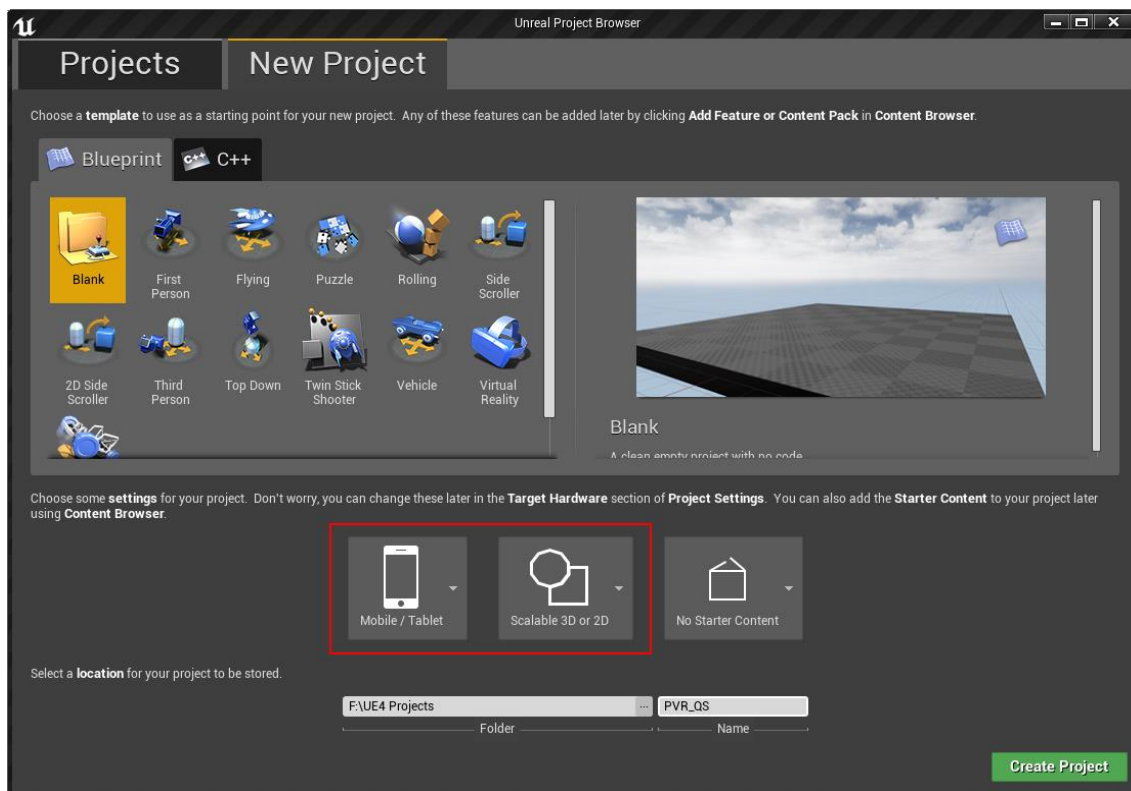


图 4.1 创建项目

请务必保证目标硬件为：Mobile/Tablet，目标图象级别为：Scalable 3D or 2D。另外项目名称及项目路径不要出现中文。

### 4.2 设置项目

为兼容我们的 SDK，需要对项目进行设置，主要包括以下几点：

- 1、设置项目的 Editor Start Map 与 Game Default Map：即保持当前地图，然后进入 Edit->Project Setting->Project->Maps and Modes，设置为项目的 Editor Start Map 与 Game Default Map 为先前保持的地图。



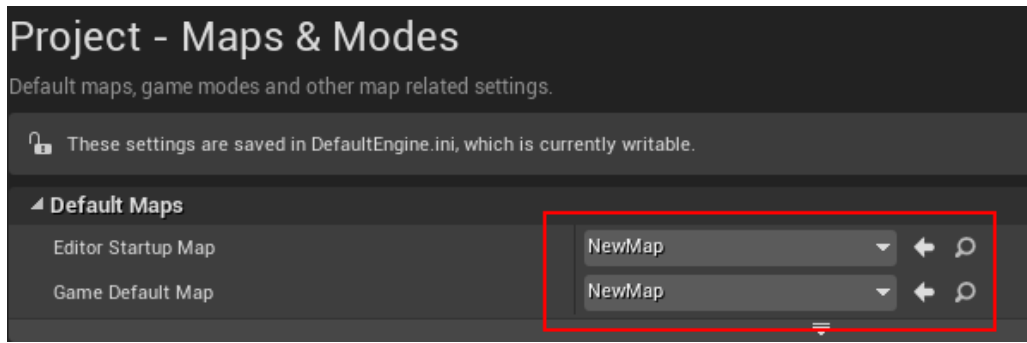


图 4.2 设置地图

2、清除默认虚拟按键：进入 Engine->Input->Mobile，清除 Default Touch Interface；

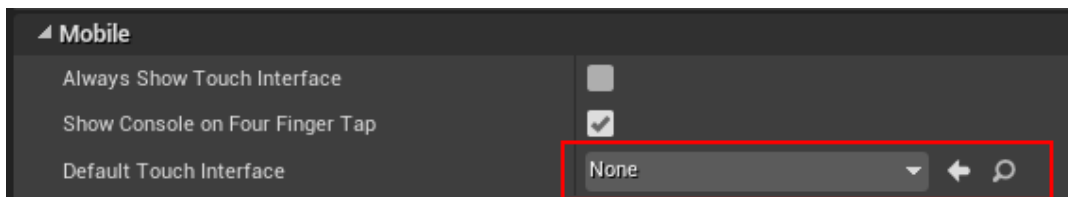


图 4.3 清除游戏默认虚拟按键

然后进入 Platforms->Android->APKPackaging，勾选 Enable FullScreen Immersive on KitKat and above devices。

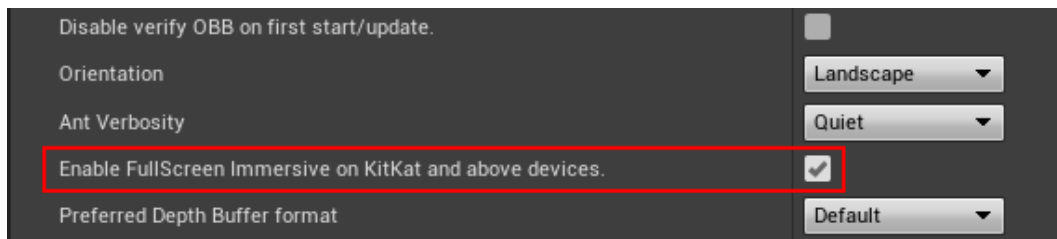


图 4.4 隐藏系统虚拟按键

3、设置 Android 平台的 SDK 与 NDK 版本：进入 Platforms->Android，将 Minimum SDK Version 与 Target SDK Version 均设为 19；

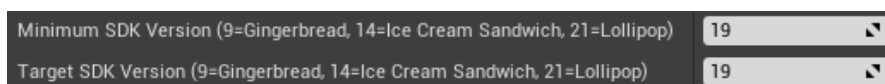


图 4.5 设置 SDK 版本

然后进入 Platforms->Android SDK，将 SDK API Level 设置为 matchndk，NDK API Level 设置为 android-19。

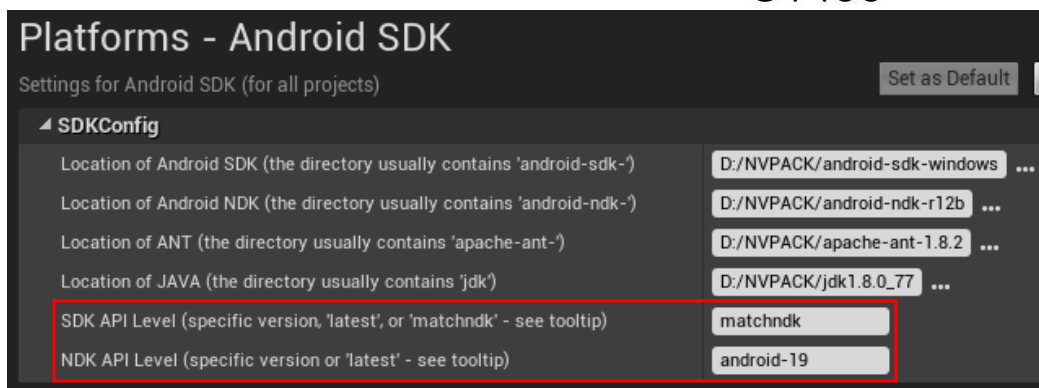


图 4.6 设置 NDK 版本

4、设置支持的 CPU 类型与图形接口类型：进入 Platforms->Android->Build，确保 Support armv7 被勾选。

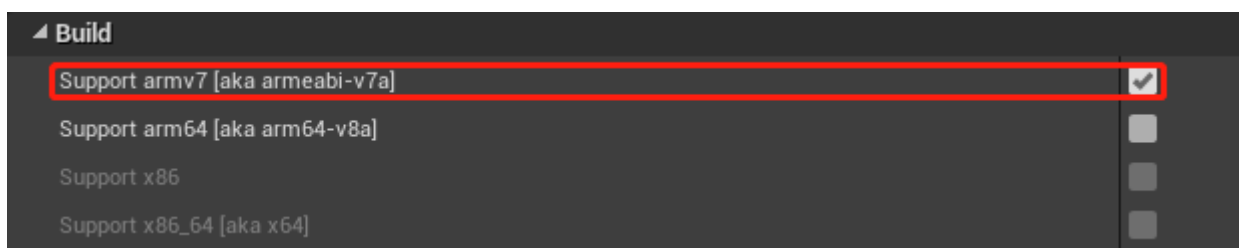


图 4.7 设置支持的 CPU 类型

5、关闭与 SDK 相冲突的 Plugins：进入 Edit->Plugins，取消勾选 Built-in / Virtual Reality 下的所有项，以及 Built-in / Input Devices 下的所有项。**此时编辑器会提示重启，我们直接关闭编辑器即可。**

### 4.3 导入 SDK 开发包

**请务必保持编辑器关闭**，将解压出的 Plugins 目录复制到项目的根目录下：

名称	修改日期	类型	大小
Config	2018/3/28 16:10	文件夹	
Content	2018/3/28 16:10	文件夹	
Intermediate	2018/3/28 16:12	文件夹	
<b>Plugins</b>	2018/3/28 16:16	文件夹	
Saved	2018/3/28 16:10	文件夹	
MyProject.uproject	2018/3/28 16:10	Unreal Engine Project File	1 KB

图 4.8 将 SDK 导入目录

然后双击项目名称重新打开。如果是 UE4.20+，会提示 SDK 模块没有编译，点击“是(Y)”即可：

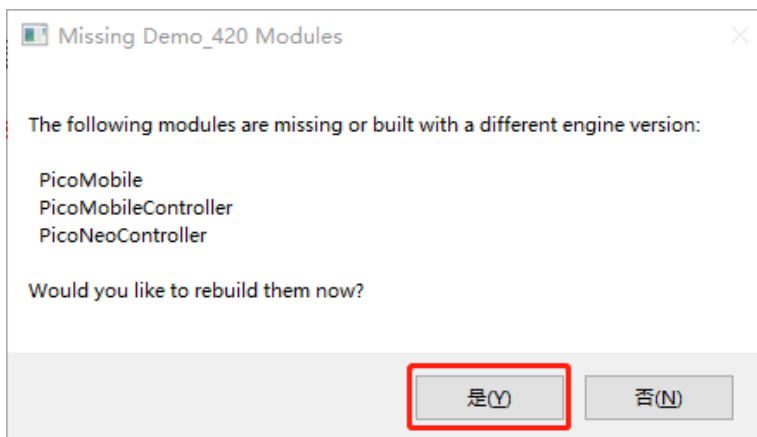


图 4.9 未编译提示

待项目打开后，在 Plugins 页面下可以看到我们的插件：

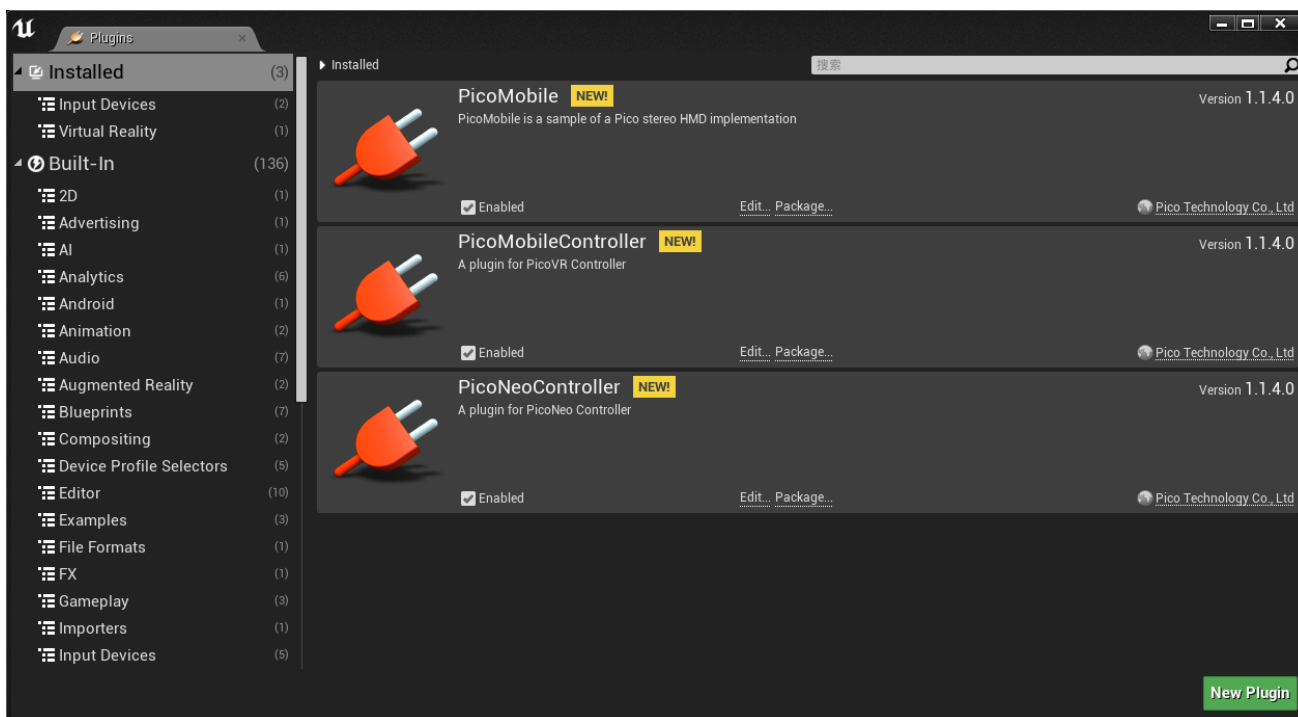


图 4.10 Pico 相关插件

## 4.4 完成项目

1、新建 Pawn 蓝图类，为其 DefaultSceneRoot 组件下添加 Scene 组件，然后在 Scene 组件下新建 Camera 组件：

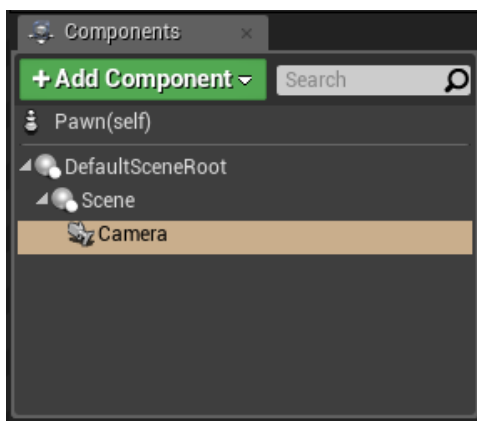


图 4.11 Pawn 组件结构

打包安装到头盔后，此 Camera 的相对位置/姿态将根据头盔实时刷新，完成头部跟踪与立体渲染。

2、将 Pawn 拖入场景，并将其 Auto Possess Player 设为 Player0:

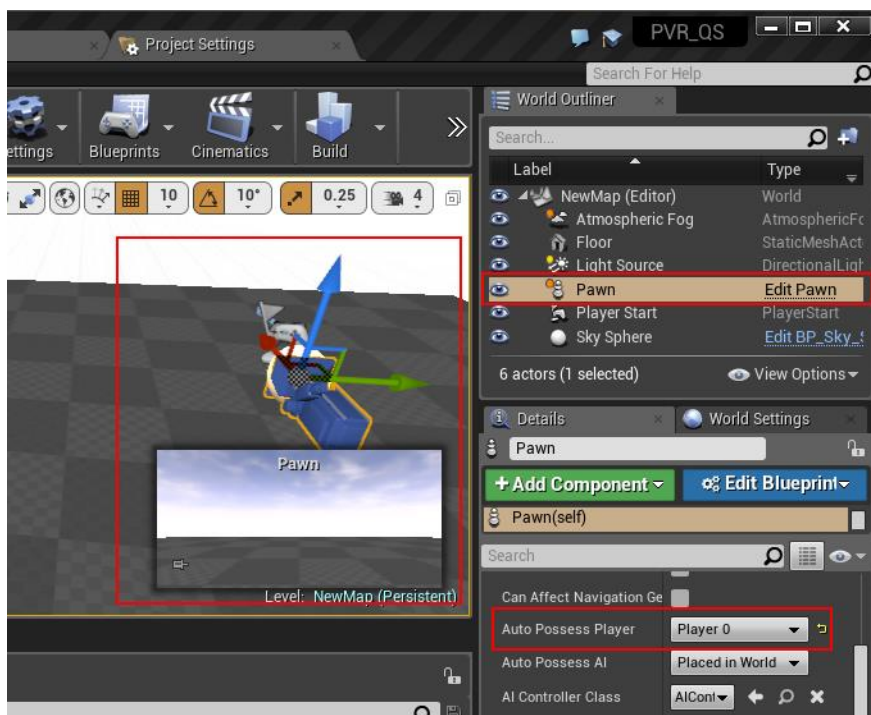


图 4.12 将 Pawn 拖入场景

## 4.5 项目打包

Pico Neo、Pico Goblin 支持的纹理压缩格式为 ASTC，所以在打包项目时需选择 Android (ASTC)。具体打包流程为：在编辑器中，执行文件->打包项目->Android-> Android (ASTC)，即可打包 (建议先进入“Project Settings”，勾选 Platforms 子项 Android 中的“Package game data inside .apk?”，以便将数据打包进 apk)：

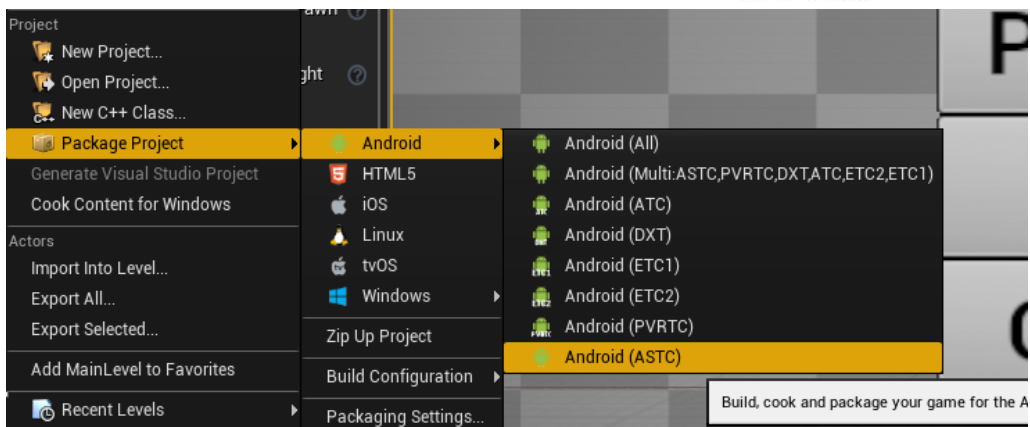


图 4.13 打包流程

另外，SDK 目前暂不支持 Gradle 构建，打包前请务必取消勾选 Enable Gradle instead of Ant:

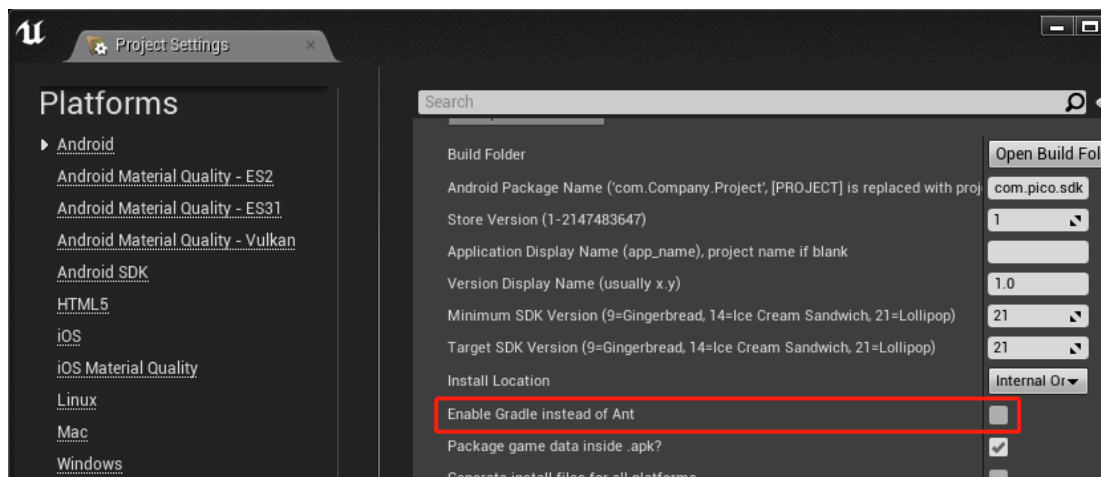


图 4.14 取消勾选 Enable Gradle instead of Ant

打包后，双击 “Install\_项目名称\_编译配置-armv7-es2.bat 进行安装”：

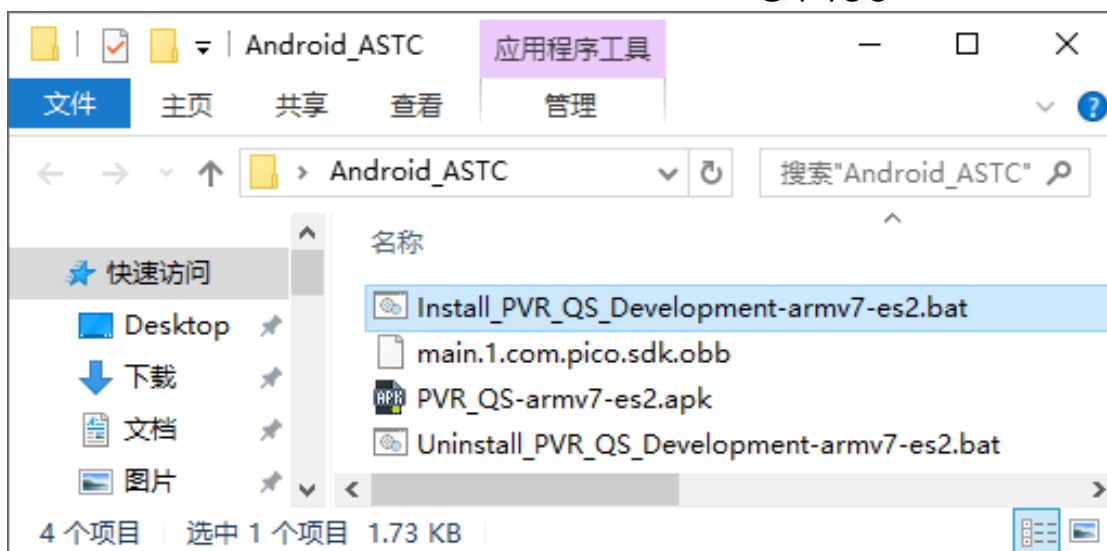


图 4.15 安装 apk

## 5 控制器使用指南

### 5.1 Pico Goblin 控制器

#### 5.1.1 使用指南



图 5.1 Pico Goblin 控制器

使用 Pico Goblin 控制器，请遵循如下步骤：

- 1、给游戏中的默认 Pawn 类添加 MotionController 组件，使其与 Camera 组件同级：

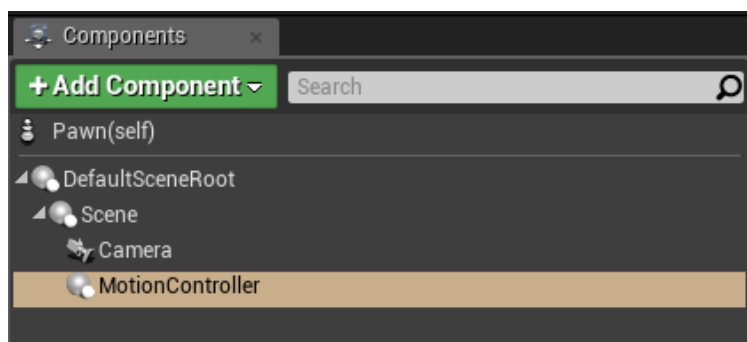


图 5.2 添加 MotionController 组件

该组件不受 Hand 属性的影响，如更换左右手，请去系统设置中修改。另外，请勾选 Disable Low Latency Update，以避免每一帧二次刷新照成的模型闪烁：

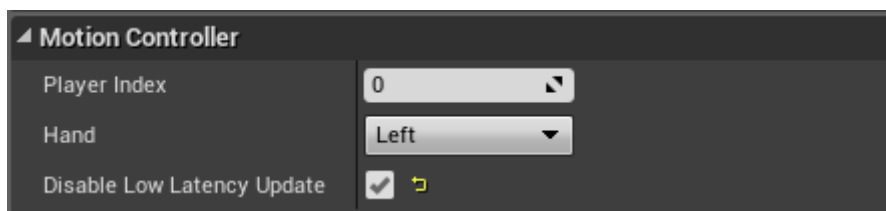


图 5.3 勾选 Disable Low Latency Update

打包安装后，该组件将跟随 Pico Goblin 手柄的位置/姿态运动。

## 2、为 MotionController 添加模型：

对于 UE4.18，请首先给 MotionController 组件添加 StaticMesh 组件：

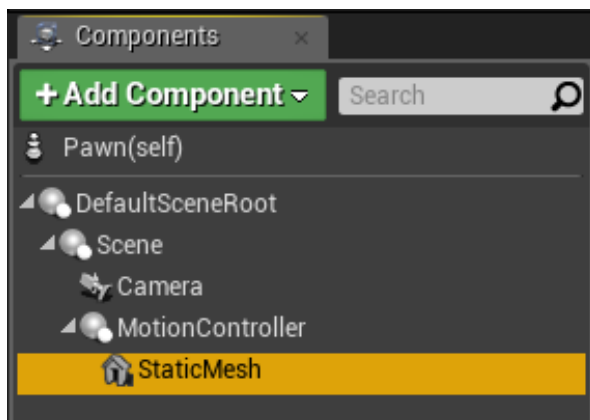


图 5.4 添加 StaticMesh 组件

添加模型时，请进入 StaticMesh 组件的细节面板，找到 StaticMesh 属性。如需添加 Pico Goblin 控制器模型，请先勾选“Show Plugin Content”，然后选择 ppcontroller：

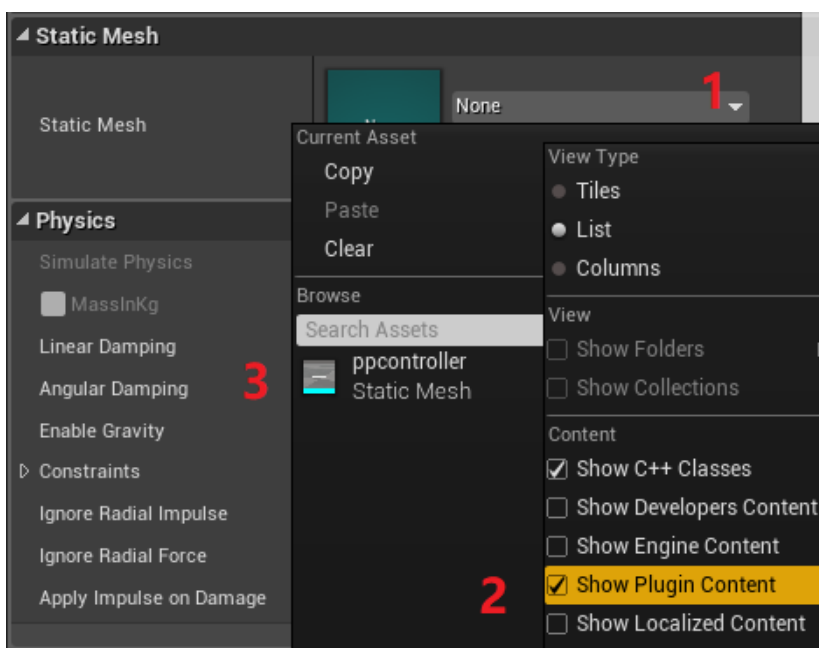


图 5.5 UE4.18 添加 Pico Goblin 控制器模型

对于 UE4.19 及更高版本，请进入 MotionController 细节面板的 Visualization 子项下添加模型（同样需勾选“Show Plugin Content”方可显示）：



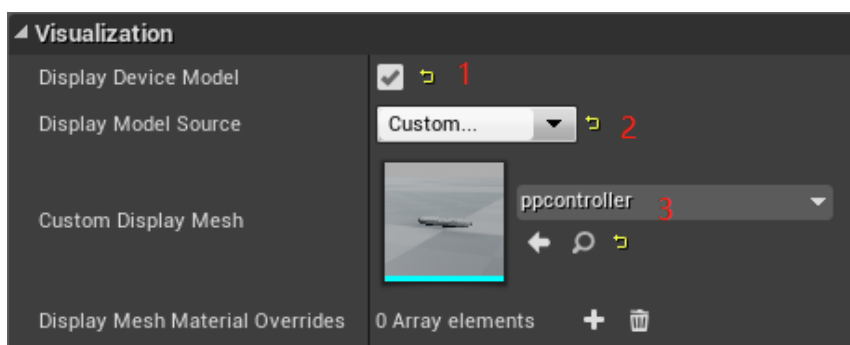


图 5.6 UE4.19+添加 Pico Goblin 控制器模型

需要说明的是，在我们的插件目录下，我们将带按键动画的手柄封装成了 Actor，如想复用请将其附加到您关卡中的 Pawn 或 Character 上。

## 5.1.2 输入说明

### 5.1.2.1 按键

Goblin 控制器所用按键并非引擎预定义的按键，开发时请根据下表设置输入绑定，或通过相应的蓝图节点驱动事件：

Goblin 控制器按键	输入绑定	蓝图节点
 <ul style="list-style-type: none"> <li>✓ APP按钮</li> <li>✓ 触摸板、OK按钮</li> <li>✓ Home 按钮</li> </ul>		
 <ul style="list-style-type: none"> <li>✓ APP按钮</li> <li>✓ 触摸板、OK按钮</li> <li>✓ Home 按钮</li> </ul>		

Goblin 控制器按键	输入绑定	蓝图节点
 <ul style="list-style-type: none"> <li>✓ APP按钮</li> <li>✓ 触摸板、OK按钮</li> <li>✓ Home 按钮</li> </ul>	<input type="text" value="Pico Handle Home"/>	
 <p>提高音量 降低音量</p>	<input type="text" value="Pico Handle VolumeUp"/>	
 <p>提高音量 降低音量</p>	<input type="text" value="Pico Handle VolumeDown"/>	

### 5.1.2.2轴

Goblin 控制器的轴输入仅含触摸板，触摸板的示意图如下：

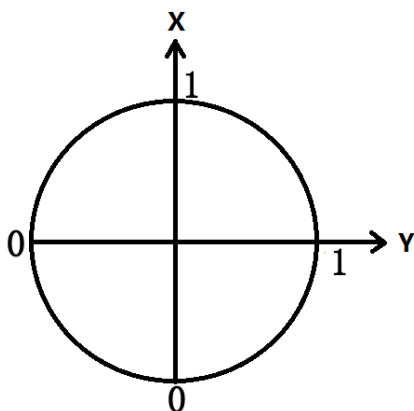


图 5.7 Goblin 控制器示意图

注意，与 Steam、Oculus、Google 等不同，我们的触摸范围是 0~1，如您的项目中有涉及输入轴的逻辑，

需要进行 0~1 到 -1~1 的映射。

通过如下节点可获得相应的输入值：



图 5.8 获取触摸板轴值的方法

### 5.1.3 相关蓝图节点

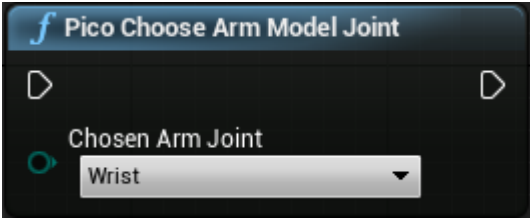
对于 Goblin 控制器，我们还提供了如下蓝图接口：

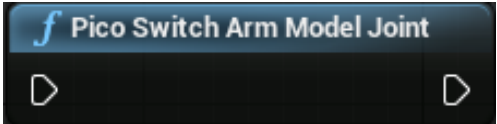
	<p>功能 设置手柄连接断开及重连的响应事件</p> <hr/> <p>输入</p> <p>OnPicoControllerConnected: 重连时的响应事件</p> <p>OnPicoControllerDisconnected: 断开时响应事件</p>
	<p>输出 无</p>
	<p>返回值 无</p>

	<p>功能 选择控制器的惯用手</p> <hr/> <p>输入 Left-左手, Right-右手</p>
	<p>输出 无</p>
	<p>返回值 无</p>

	<p>功能 切换控制器的惯用手</p> <hr/> <p>输入 无</p>
--	---------------------------------------

输出	无
返回值	无

	功能	选择运动控制器的当前跟踪关节
	输入	Wrist-手腕, Elbow-手肘, Shoulder-肩膀
	输出	无
	返回值	无

	功能	循环切换运动控制器的跟踪关节, 依次为手腕、手肘、肩膀
	输入	无
	输出	无
	返回值	无

## 5.2 Pico G2 控制器



图 5.9 Pico G2 控制器

Pico G2 控制器按键对应关系如下:

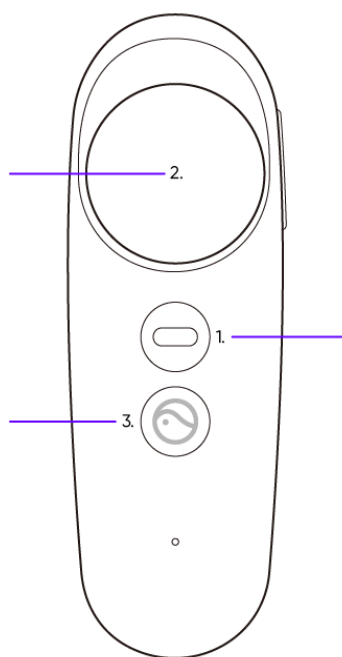


图 5.10 Pico G2 控制器按钮对应图 1

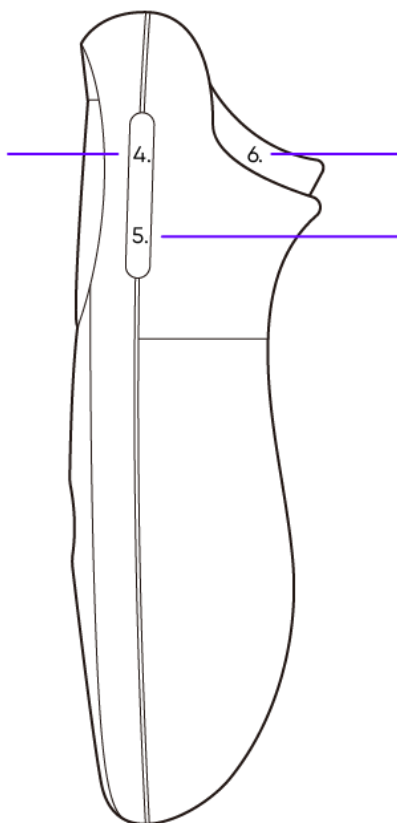


图 5.11 Pico G2 控制器按钮对应图 2

G2 控制器按键	输入绑定	蓝图节点
1、APP 按钮		
2、触摸板、OK 按钮		
3、Home 按钮		
4、提高音量		
5、降低音量		
6、扳机按钮		

## 5.3 Pico Neo 控制器

### 5.3.1 说明

Pico Neo 控制器如下图所示：



图 5.12 Pico Neo 控制器

对于 Pico Neo 控制器，有一个非常重要的逻辑概念：主手柄、副手柄。主手柄是指系统中拥有射线可以与 UI 交互的手柄，而副手柄则是另外一个手柄。当只连接一个手柄时，当前连接的手柄即为主手柄。

也建议开发者将 `WidgetInteraction` 组件附加到主手柄上，以让应用中通过射线与 UI 交互的手柄，与系统中通过射线与 UI 交互的手柄一致。

### 5.3.2 使用指南

使用 Pico Goblin 控制器，请遵循如下步骤：

- 1、给游戏中的默认 `Pawn` 类添加两个 `MotionController` 组件，分别命名为 `MotionController_Main`、`MotionController_Sub`，使其与 `Camera` 组件同级：

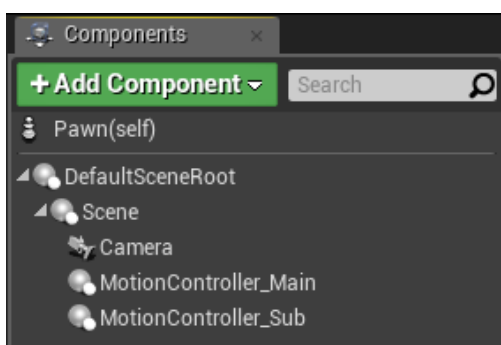


图 5.13 添加 `MotionController` 组件

选中 `MotionController_Main`，在其细节面板中找到 `Hand` 属性，将其修改为 `Special 1`，如此以来该组件将跟随主手柄运动（在只用一个手柄的情况下，请将其 `Hand` 属性设置为 `Special 1`）：

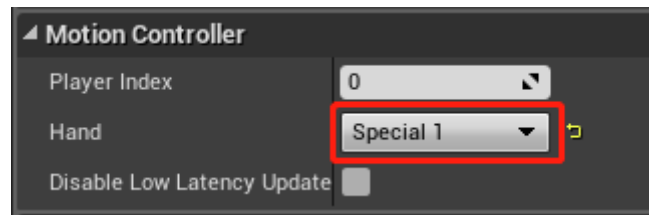


图 5.14 设置 Hand 属性

同样，对于 MotionController\_Sub，则需将其 Hand 属性设置为 Special 2。

#### 4、为 MotionController 添加模型：

对于 UE4.18，请首先给 MotionController 组件添加 StaticMesh 组件：

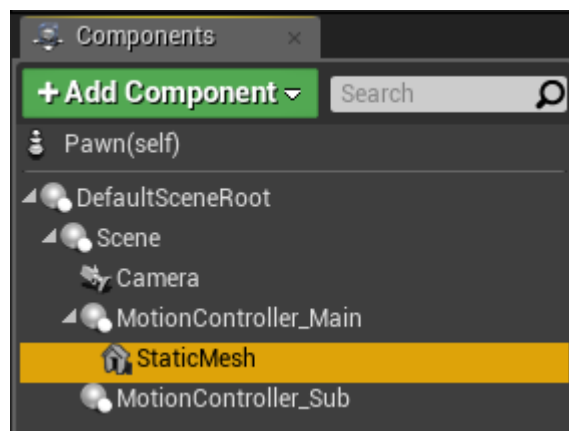


图 5.15 添加 StaticMesh 组件

然后进入 StaticMesh 组件的细节面板，找到 StaticMesh 属性。如需添加 Pico Neo 控制器模型，请先勾选“Show Plugin Content”，然后选择 Mesh\_cvcontroller：

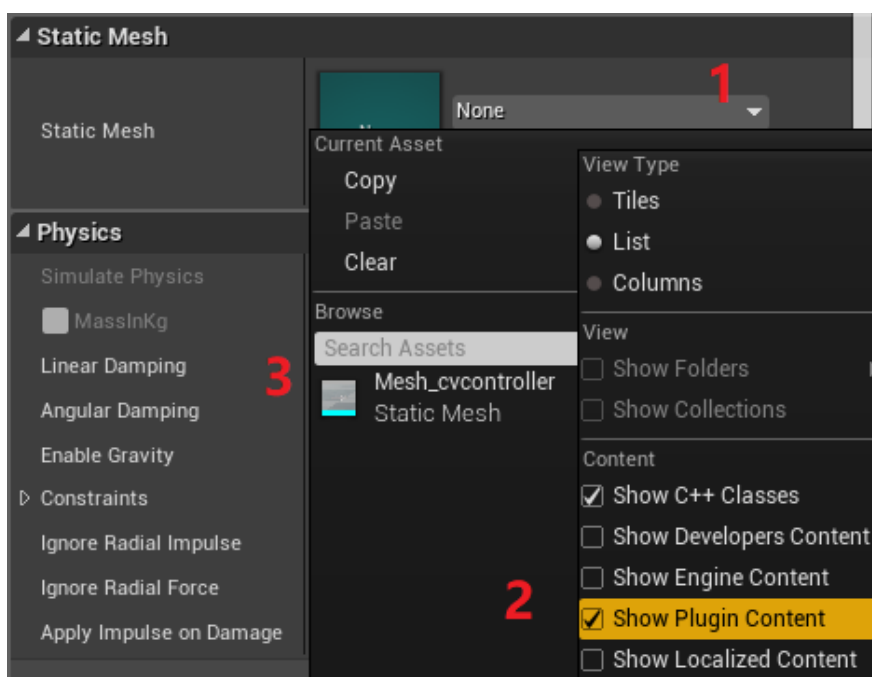




图 5.16 添加 Pico Goblin 控制器模型

对于 UE4.19 及更高版本，请进入 MotionController 细节面板的 Visualization 子项下添加模型（同样需勾选“Show Plugin Content”方可显示）：

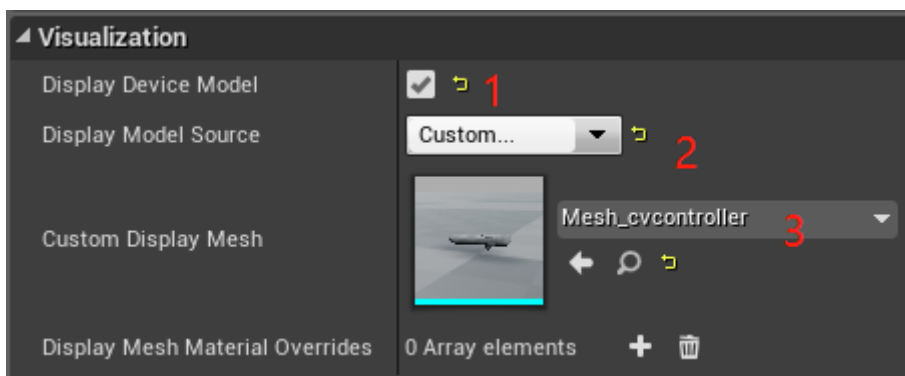




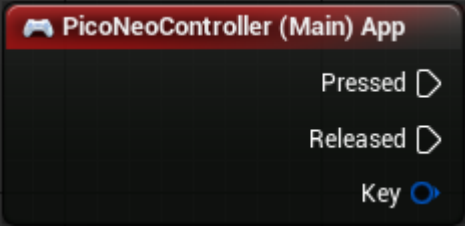

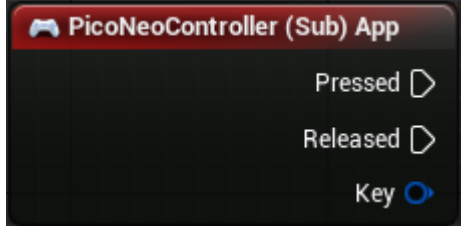









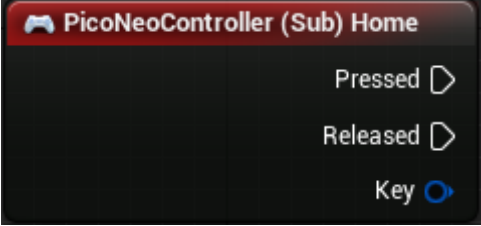
图 5.17 UE4. 19+ 添加控制器模型

需要说明的是，在我们的插件目录下，我们将带按键动画的手柄封装成了 Actor，如想复用请将其附加到您关卡中的 Pawn 或 Character 上。

### 5.3.3 输入说明

#### 5.3.3.1 按键

Neo 控制器所用按键并非引擎预定义的按键，开发时请根据下表设置输入绑定，或通过相应的蓝图节点驱动事件：

按键	输入事件		
	主手		
	副手		
	主手		
	副手		
	主手		
	副手		

按键	输入事件		
	主手		
	副手		
	主手		
	副手		

### 5.3.3.2轴

Pico Neo 控制器的轴输入分为触摸板与扳机，触摸板的示意图如下：

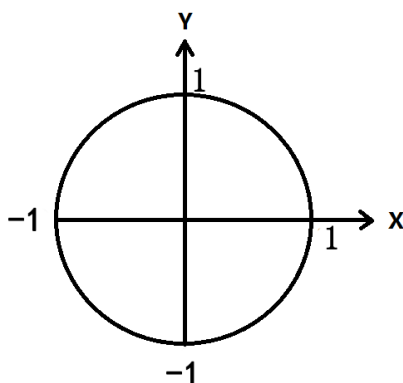


图 5.18 Pico Neo 控制器触摸板示意图

分别可通过如下蓝图节点获取主、副手柄触摸板的值：



图 5.19 触摸板相关蓝图节点

扳机的输入范围为 0~1，可通过如下蓝图节点获取主、副手柄的扳机输入值：



图 5.20 扳机相关蓝图节点

### 5.3.4 相关蓝图节点

对于 Pico Neo 控制器，我们以蓝图节点的形式提供了若干 API，在事件图表中点击鼠标右键，展开 Pico Neo->Controller，可以看到这些 API：

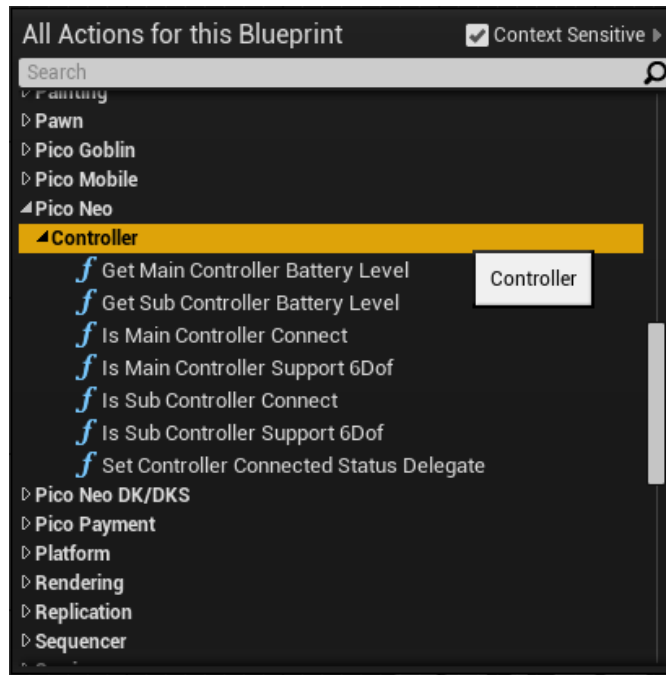




图 5.21 Pico Neo 控制器 API


这些 API 的详细用法如下：


	<table border="1"> <tr> <td>功能</td> <td>设置手柄断开、回连时的自定义事件</td> </tr> <tr> <td>输入</td> <td>                     OnMainControllerConnect: 主手柄回连                      OnMainControllerDisconnect: 主手柄断开                      OnSubControllerConnect: 副手柄回连                      OnMainControllerDisconnect: 副手柄断开                 </td> </tr> <tr> <td>输出</td> <td>无</td> </tr> <tr> <td>返回值</td> <td>无</td> </tr> </table>	功能	设置手柄断开、回连时的自定义事件	输入	OnMainControllerConnect: 主手柄回连 OnMainControllerDisconnect: 主手柄断开 OnSubControllerConnect: 副手柄回连 OnMainControllerDisconnect: 副手柄断开	输出	无	返回值	无
功能	设置手柄断开、回连时的自定义事件								
输入	OnMainControllerConnect: 主手柄回连 OnMainControllerDisconnect: 主手柄断开 OnSubControllerConnect: 副手柄回连 OnMainControllerDisconnect: 副手柄断开								
输出	无								
返回值	无								


	<table border="1"> <tr> <td>功能</td> <td>获取主手柄的电量</td> </tr> <tr> <td>输入</td> <td>无</td> </tr> <tr> <td>输出</td> <td>无</td> </tr> <tr> <td>返回值</td> <td>主手柄的电量, 1~10</td> </tr> </table>	功能	获取主手柄的电量	输入	无	输出	无	返回值	主手柄的电量, 1~10
功能	获取主手柄的电量								
输入	无								
输出	无								
返回值	主手柄的电量, 1~10								

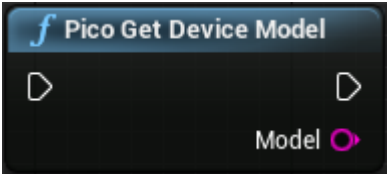
	功能	获取副手柄的电量
	输入	无
	输出	无
	返回值	副手柄的电量, 1~10

	功能	判断主手柄是否连接
	输入	无
	输出	无
	返回值	true-连接, false-未连接

	功能	判断副手柄是否连接
	输入	无
	输出	无
	返回值	true-连接, false-未连接

	功能	判断主手柄是否支持 6DoF 跟踪
	输入	无
	输出	无
	返回值	true-支持, false-不支持 (即为 3DoF 跟踪)

	功能	判断副手柄是否支持 6DoF 跟踪
	输入	无
	输出	无
	返回值	true-支持, false-不支持 (即为 3DoF 跟踪)

	功能	获取头显类型
	输入	无
	输出	代表头显类型的字符串
	返回值	无

## 6 接口函数

### 6.1 通用函数库

SDK 支持如下红点标记的引擎中的 VR 通用函数：

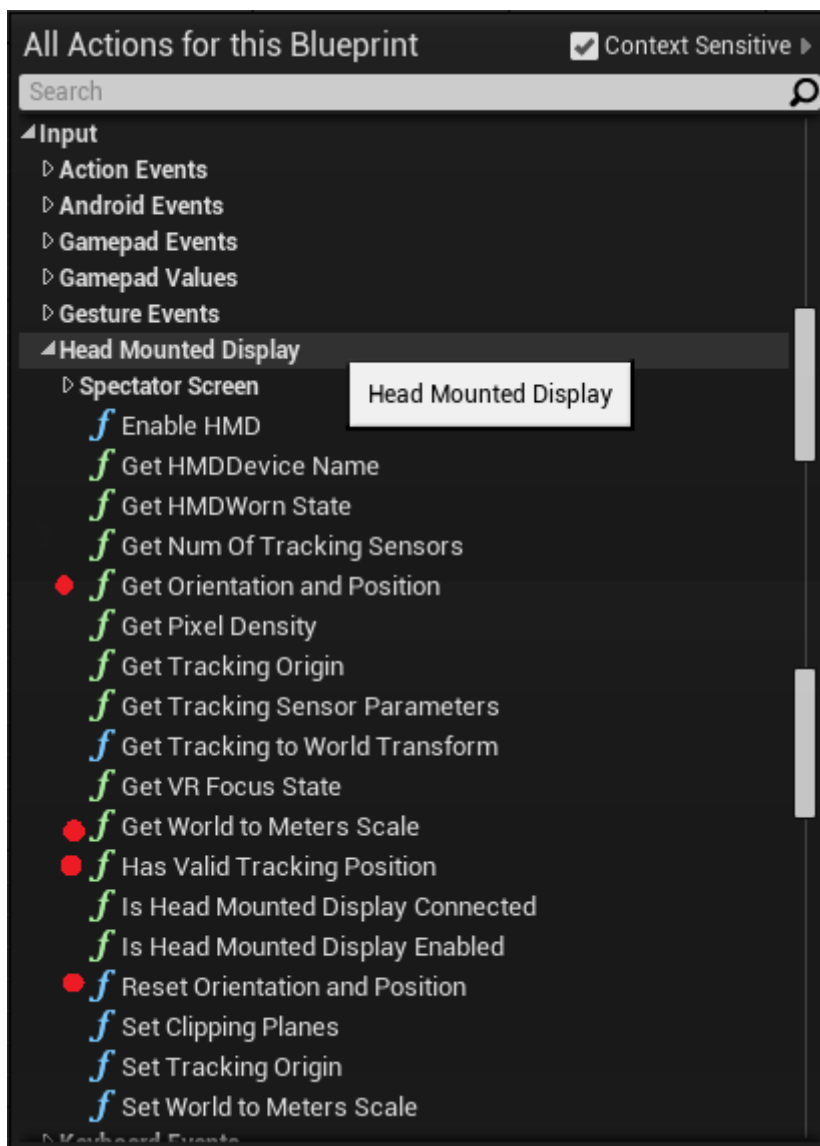


图 6.1 支持的通用函数

他们的详细用法请参考 UE4 官方文档 <https://docs.unrealengine.com/en-us/>。

其中，Reset Orientation and Position 节点，仅支持重置正方向的功能，函数节点中的 Yaw 参数值不起作用。

### 6.2 专用函数库

对于音量、亮度等系统功能，SDK 也以蓝图节点的形式提供了相应的 API，在事件图表中单击鼠标右键，进入



Pico Mobile 子项, 即可看到这些 API:

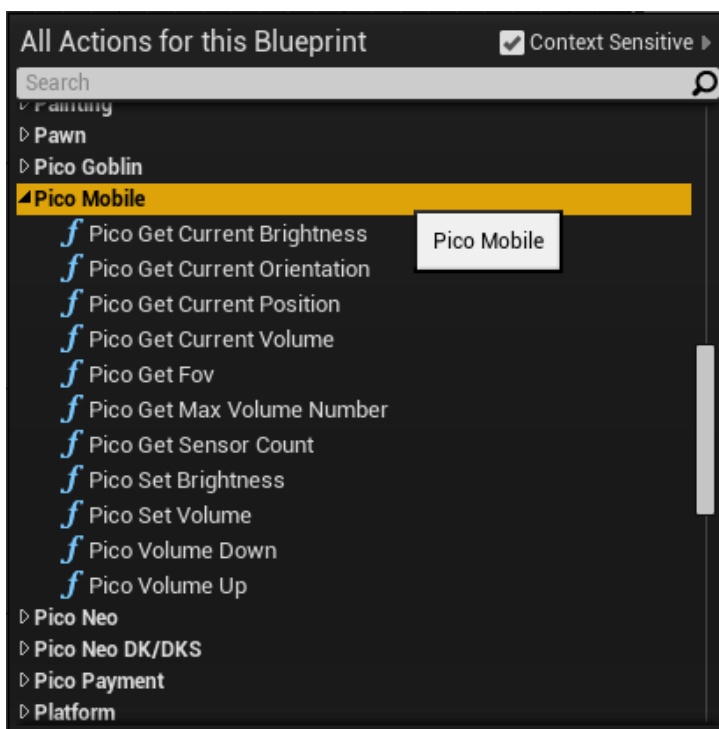
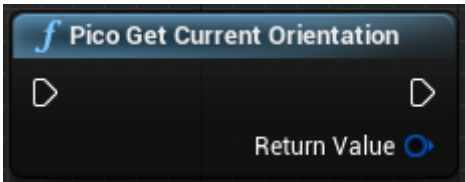
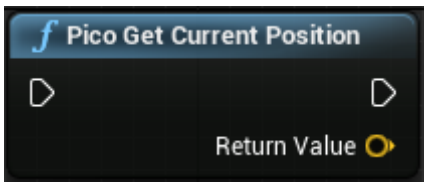
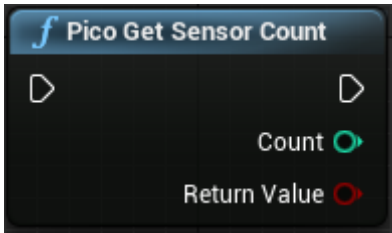


图 6.2 系统接口

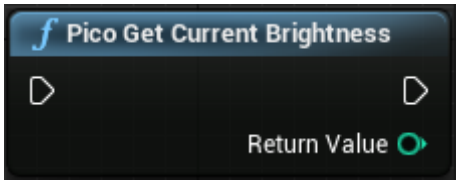
这些 API 的详细说明如下:

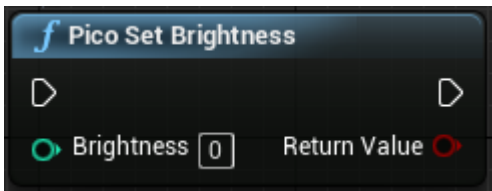
	功能	获取头戴的当前姿态
	输入	无
	输出	无
	返回值	头戴的当前姿态

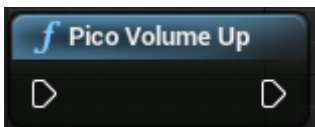
	功能	获取头戴的当前位置
	输入	无
	输出	无
	返回值	头戴的当前位置

	功能	获取当前所使用传感器的数量
	输入	无
	输出	当前所用传感器的数量
	返回值	true-成功获取, false-获取失败

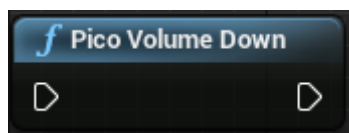
	功能	获取 FOV
	输入	无
	输出	无
	返回值	FOV

	功能	获取系统亮度
	输入	无
	输出	无
	返回值	当前系统的亮度 (0~255)

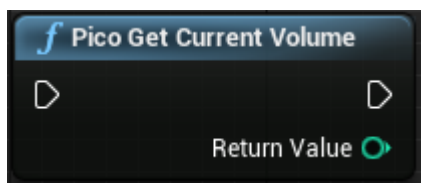
	功能	设置系统的亮度
	输入	设置的目标亮度 (0~255)
	输出	无
	返回值	true-设置成功, false-设置失败

	功能	增大系统音量 (系统音量范围为 0~15, 调用一次增加 1)
	输入	无

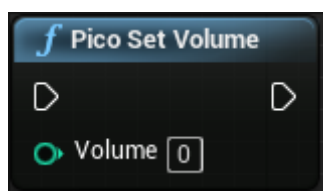
输出	无
返回值	无



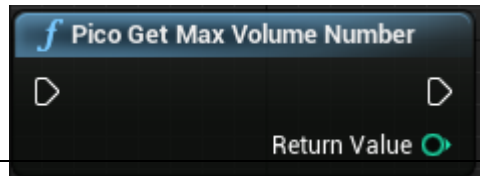
功能	减小系统音量 (系统音量范围为 0~15, 调用一次减少 1)
输入	无
输出	无
返回值	无



功能	获取当前的系统音量
输入	无
输出	无
返回值	当前的系统音量

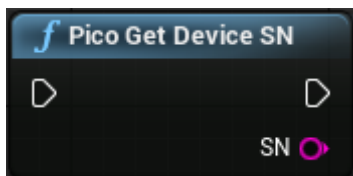


功能	设置系统音量
输入	想要设置的系统音量 (0~15)
输出	无
返回值	无

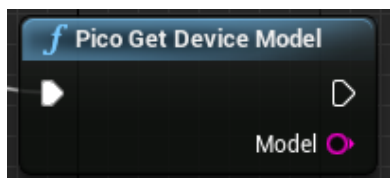


功能	获取最大音量
输入	无

输出	无
返回值	表示最大音量的数



功能	获取设备的序列号
输入	无
输出	设备序列号
返回值	无



功能	获取设备的 Model
输入	无
输出	设备 Model (Pico Neo --- Pico Neo) (Pico Goblin--- Pico Goblin) (Pico Goblin 2--- Pico G2)
返回值	无

## 7 支付系统

Pico 支付, 是基于 Pico 账户体系进行的游戏币支付系统, 结算方式以现行的 Pico 公司下的游戏货币单位为准 (P 币)。

## 7.1 准备工作

### 7.1.1 申请并填入 APPKEY、APPID、SCOPE、DEVELOPERID、APP SECRET

开发者在接入支付 SDK 时，需要在开发者平台创建应用并获取相应字符串。申请流程如下：

- 1. 登录开发者平台并注册 Pico 会员 (<http://dev.picovr.com/>)<sup>1</sup>
- 2. 申请成为开发者

开发者分为个人开发者和企业开发者，请根据实际情况进行申请。审核提交后，我们会在 3 个工作日内进行反馈，请及时查看开发者平台状态。

- 3. 查看商户 ID

申请成为开发者后，点击右上角昵称可以查看到开发者 ID，即商户 ID：



图 7.1 开发者 ID

- 4. 获取相应字符串

开发者可以从“应用管理”进入到“创建应用”阶段：

---

<sup>1</sup> 注：在申请开发者前，需要先成为 Pico 会员。

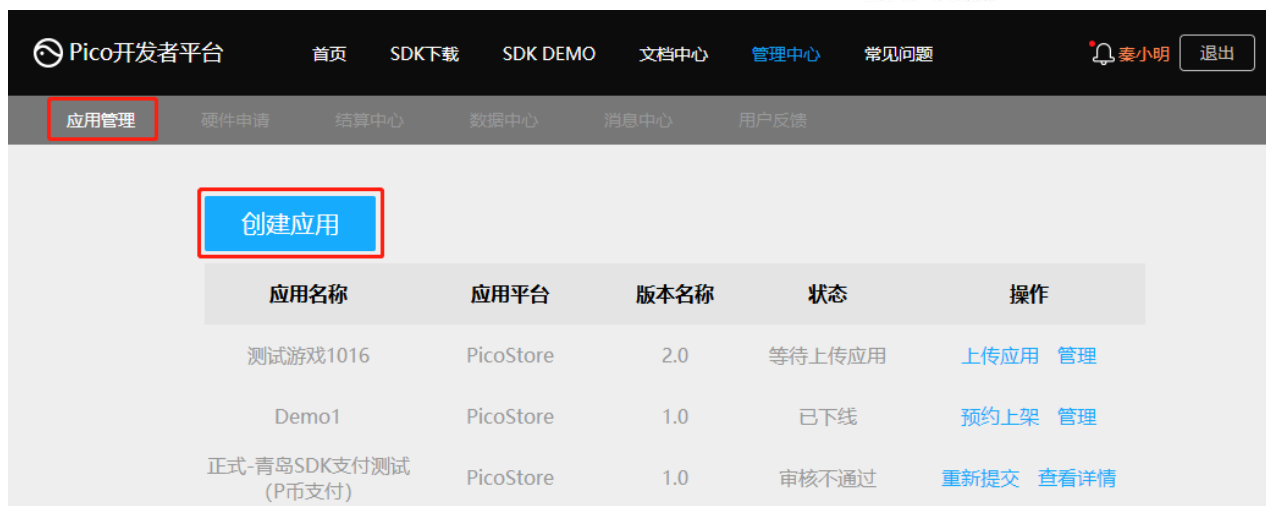


图 7.2 创建应用

点击创建应用后，需要选择要发布的平台：



图 7.3 选择应用的发布平台

选择平台后即可完善应用的相关信息，包括应用是免费还是付费，付费时需要支付多少 P 币：

基础信息

编辑资料

中文  
英文  
日文

应用名称：应用名称

应用简介：简单描述应用

应用介绍：不超过400字

应用截图：

文件格式为jpg或png；尺寸为800\*450像素  
PicoStore应用列表将使用您上传的首张截图用作展示

应用类型： 应用  游戏 提示：应用类型一经选择无法修改

图 7.4 完善应用的相关信息

注：请重点关注上图标红位置，请谨慎填写应用类型，一经填写是无法进行修改的！成功创建应用后，开发者平台会对其分配字符串，包括 APPKEY，APPID、APP SECRET：

APP名称：aila

APPID：2bd5d117f53b491d64d3b9cf21fd07c8

APP KEY：4ef7d5a7aba7bf5e8c57e3c2394ba88f

APP Secret：0f6e554c6b3c8903351f2b2911894a50

状态：等待上传应用

提交时间：2017-04-14 13:46:31

操作：[查看信息](#)

[上传应用](#) [游戏内支付配置](#)

提示：游戏内支付配置仅针对“游戏类型”生效！请在上传应用前先进行道具配置，无内付游戏请忽略。

图 7.5 APP ID、APP KEY、APP Secret

游戏类应用如果存在道具内付的情况，我们要求开发者必须采用开发者后台增加商品码的方式进行统一管理。  
请选择“游戏内支付配置”，配置游戏的内购信息：



图 7.6 游戏内购配置

注意，商品码的规则定义为‘首位为字母，仅允许输入字母及数字，不超过 20 位字符’。不同道具间的商品码不能重复。道具类型分为可消耗道具和不可消耗道具。可消耗道具为可重复购买的商品，如金币、血瓶等；不可消耗道具为一次性购买产品，如武器、解锁关卡。

#### ➤ 5. 填入字符串

进入 Edit->Project Settings..., 展开 Plugins 子项下的 PicoMobile, 勾选“Enable Payment Module”, 然后根据实际情况勾选“Is Foreign”, 最后将获取的商户（开发者）ID、APPID、APP KEY、APP secret 填入以下位置：



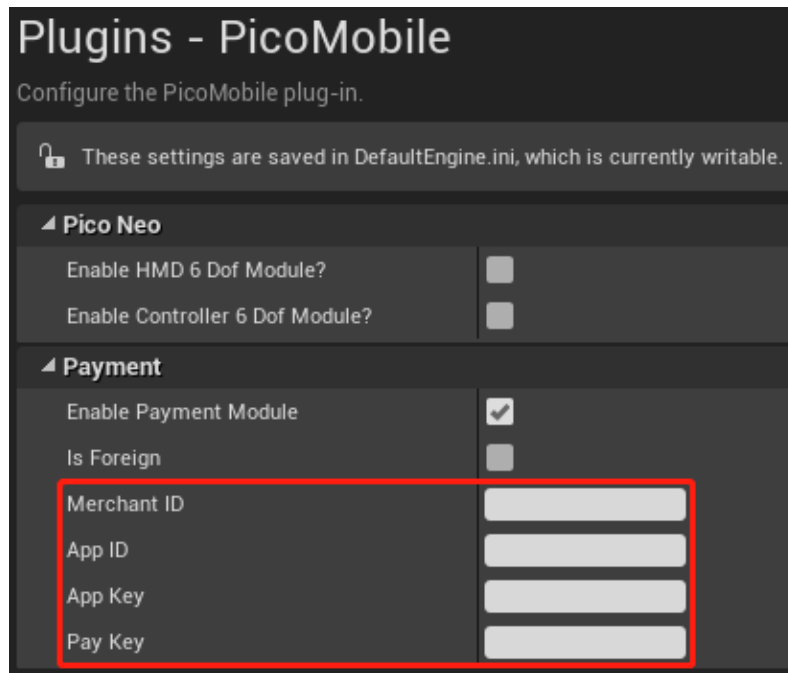
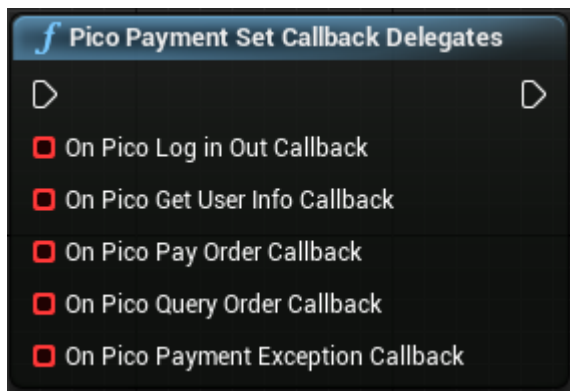


图 7.7 填入字符串

### 7.1.2 设置回调代理事件

使用支付之前，首先要设置回调代理事件，这样您便可以获取回调函数输出的参数，并设置后续执行流程。这

里请使用我们提供的 PicoPaymentSetCallbackDelegates 节点：

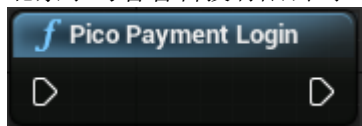


其中，On Pico Payment Exception Callback 为支付产生各种异常的回调，至于其余各回调函数参数的确切含义，将在下一节介绍其相关的主调函数时介绍。

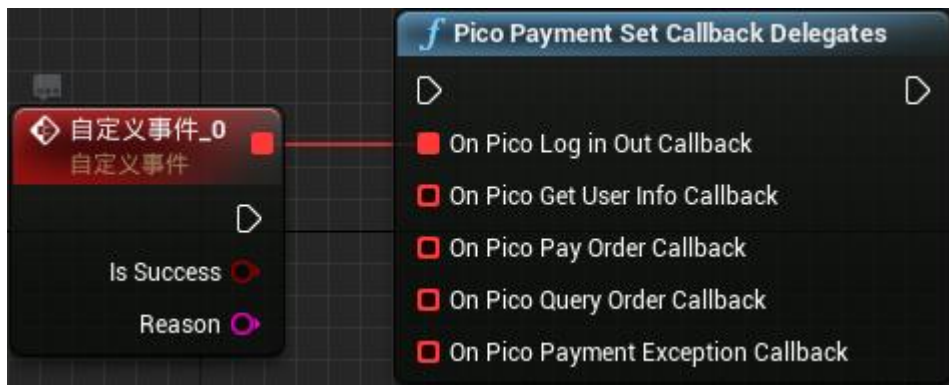
### 7.1.3 用户登录

Pico 为开发者提供基于 Oauth2.0 模式的认证授权，故用户支付前需要先进行登录操作，这里使用我们提供的

PicoPaymentLogin 节点：



- 回调函数: OnPicoLogInOutCallback, 其参数如下:

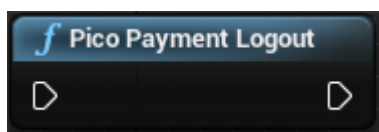


- IsSuccess: 登录、登出是否成功 (boolean) , true 表示成功, false 表示失败
- Reason: 登录、登出成功或失败的原因

登陆部分可以只登陆一次, 之后直接使用支付即可, 登陆过期时间约为两周, 过期后支付接口回有返回码 (登陆过期码), 用户只需再次登陆即可。

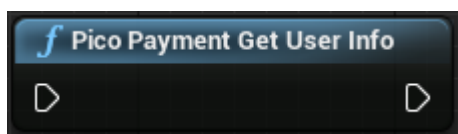
## 7.2 其他相关接口

### 7.2.1 PicoPaymentLogout



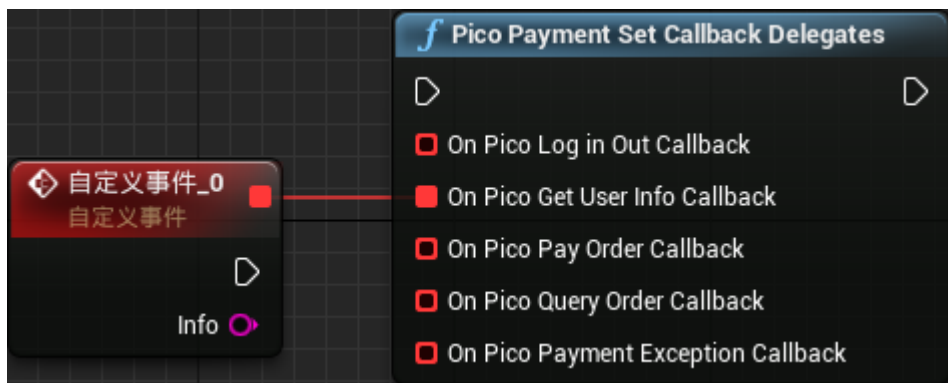
- 函数功能: 登出
- 其回调函数: OnPicoLogInOutCallback 已在上文介绍。

### 7.2.2 PicoPaymentGetUserInfo



- 函数功能: 获取用户信息

➤ 回调函数 OnPicoGetUserInfoCallback:



➤ Info: 一个未经处理的 Json 串 (string) , 查询成功范例如下:

```
{
  "ret_code": "0000",
  "data": {
    "aboutme": "",
    "birthday": 1460476800000,
    "phone": "13100000000",
    "username": "Admin",
    "email": "",
    "gender": "male",
    "lastname": "",
    "openid": "4f3148bdc34d9bca104927729a173b64",
    "firstname": "",
    "avatar": "http://172.31.83.11/upload/6dd6ee103714e967846c3d38ae48d511",
    "signature": "14a25d7219d8dfc91e55f63286ae5c0a",
    "country": "China",
    "city": ""
  },
  "ret_msg": "调用成功"
}
```

查询失败范例如下:

```
{
  "ret_code": "00003000",
  "ret_msg": "签名验证失败"
}
```

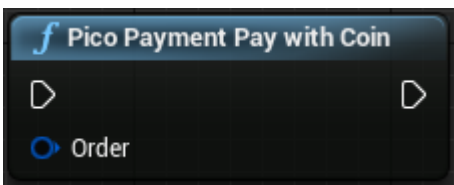
其他 ret\_code 码及 ret\_msg 一览:

表 7.1 OnPicoGetUserInfoCallback 输出参数 ret\_code 码及 ret\_msg 一览

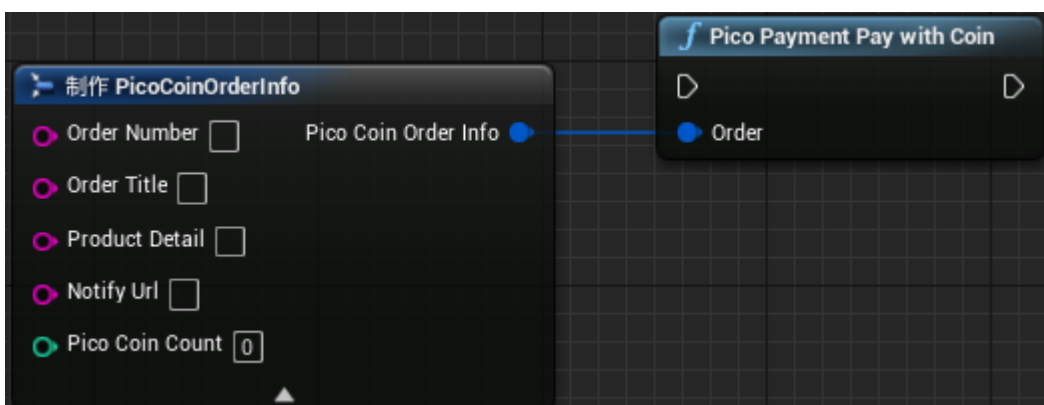
ret_code	ret_msg
----------	---------

ret_code	ret_msg
0000	请求成功
00020000	数据库操作失败
9999	系统错误
00001000	参数错误
00002000	数据解析失败
00003000	签名验证失败
00003001	时间验证失败
00060000	用户未找到
00060001	用户密码错误
00060002	用户登录未知错误
00061000	用户 token 未找到失败
00061001	用户 token 验证失败
00061002	用户 token 未知错误
00070001	应用验证失败
00071001	应用密钥验证失败
00080001	OAUTH_CODE 验证失败
00090001	REFRESH_TOKEN 验证失败
00100001	ACCESS_TOKEN 验证失败
00110001	SCOPE 验证失败

### 7.2.3 PicoPaymentPaywithCoin

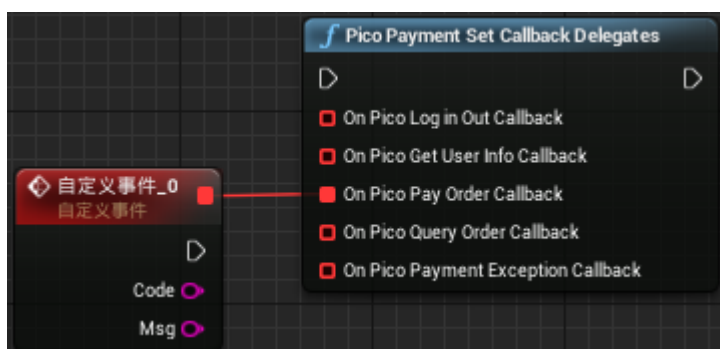


- 函数功能：使用 P 币支付
- 输入参数 Order:



- OrderNumber: 商户自己定义的订单号,32 个字符内、可包含字母和数字;

- OrderTitle: 订单标题;
- ProductDetail: 商品描述;
- Notify Url: 欲通知的 URL (非必填) , **必须为直接可访问的 url, 不能携带参数;**
- PicoCoinCount: 花费 P 币数额。
- 回调函数 OnPicoPayOrderCallback:

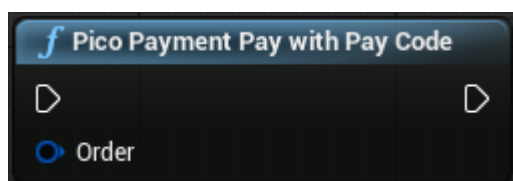


Code 及 Msg 如下:

Code	Message 信息
00000	网络异常
10000	登录成功
10001	用户未登陆
10002	请输入正确金额
10003	登陆过期, 请重新登陆
11000	商户验证成功
11001	商户验证失败
11002	用户验证参数错误或请求过期
11003	商户未验证
12000	支付成功
12001	支付失败
12003	P 币不足
12004	余额可用
13000	生成订单
13001	获取数据失败
13002	生成订单失败

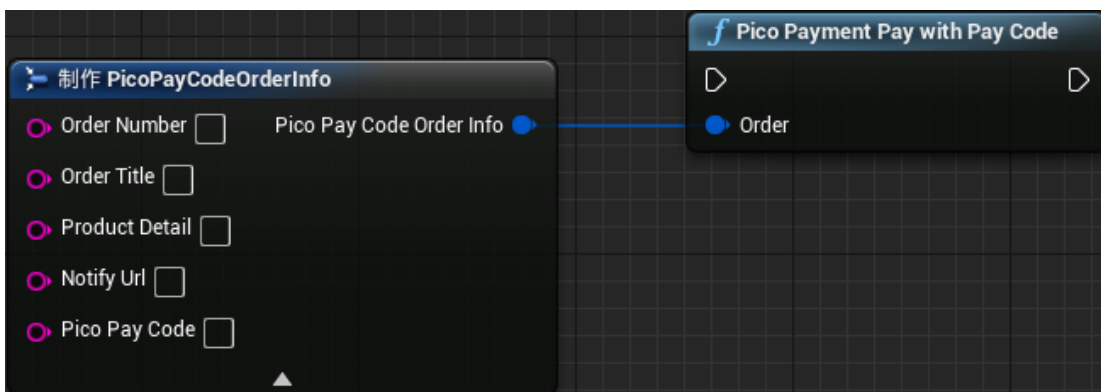
Code	Message 信息
14000	查询订单成功
14001	订单不存在/有误
14002	用户取消支付操作
15000	未输入商品信息
15001	未输入预付 ID
15002	请输入 Pico 支付订单号或商户订单号
NOAUTH	商户无此接口权限
SYSTEMERROR	系统错误
APP_ID_NOT_EXIST	APP_ID 不存在
MCHID_NOT_EXIST	MCHID 不存在
APP_ID_MCHID_NOT_MATCH	app_id 和 mch_id 不匹配
LACK_PARAMS	缺少参数
SIGNERROR	签名错误
NO_DATA	没有查询到数据/用户未充值
ORDER_EXIST	订单已存在
PAY_CODE_NOT_EXIST	消费代码不存在
PAY_CODE_EXIST	用户已对商品代码消费

#### 7.2.4 PicoPaymentPayWithPayCode<sup>2</sup>



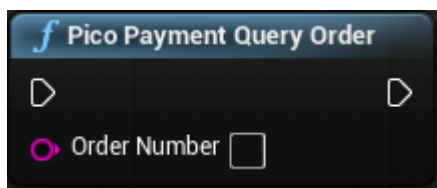
- 函数功能：使用支付码支付
- 输入参数 Order:

<sup>2</sup> 注：商品码支付是开发者平台的专为游戏设计的新支付方式，开发者需要在开发者平台自己的游戏下创建不同的商品，并填写所属的商品码。在游戏的进行开发时，不必填写物品金额，直接填写对应的商品码即可调用对应的支付接口进行支付。

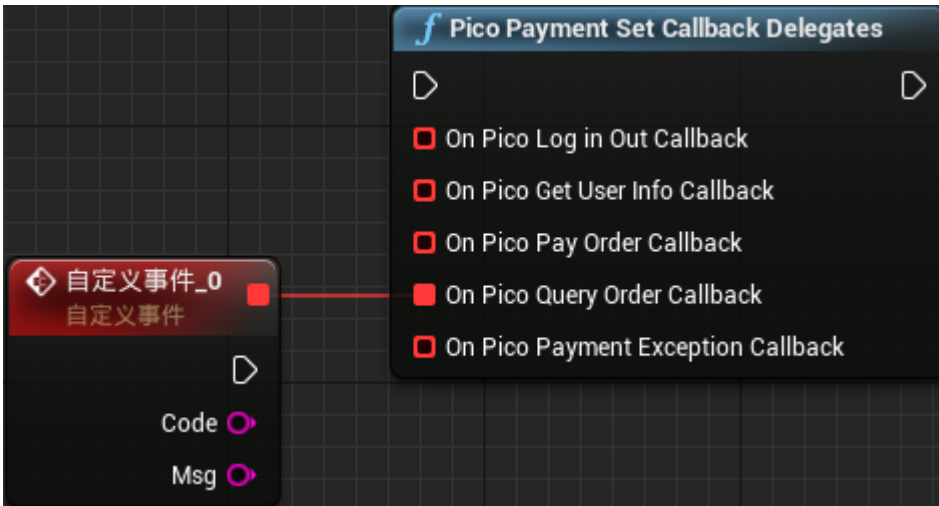


- OrderNumber: 商户自己生成的订单号,32 个字符内、可包含字母和数字;
- OrderTitile: 订单标题;
- Product Detail: 商品描述;
- Notify Url: 欲通知的 URL (非必填) , **必须为直接可访问的 url, 不能携带参数;**
- PicoPayCode: 即商品代码, 用户通过 7.1.1 游戏内支付配置获取。
- 回调函数: OnPicoPayOrderCallback, 同 P 币支付。

### 7.2.5 PicoPaymentQueryOrder



- 函数功能: 查询订单
- 输入参数 OrderNumber: 订单号 (String)
- 回调函数 OnPicoQueryOrderCallback:



➤ 参数含义同 OnPicoPayOrderCallback

### 7.3 开发者服务端交互

支付完成后，支付系统会把相关支付结果和用户信息发送给商户，商户需要接收处理，并返回应答。

对后台通知交互时，如果支付系统收到商户的应答不是成功或超时，则认为通知失败，支付系统会通过一定的策略定期重新发起通知，尽可能提高通知的成功率，但不保证通知最终能成功。

同样的通知可能会多次发送给商户系统，商户系统必须能够正确处理重复的通知。推荐的做法是，当收到通知进行处理时，首先检查对应业务数据的状态，判断该通知是否已经处理过，如果没有处理过再进行处理，如果处理过直接返回结果成功。在对业务数据进行状态检查和处理之前，要采用数据锁进行并发控制，以避免函数重入造成的数据混乱。

商户服务端需要实现下面的接口，用于接收来自 Pico 服务器请求，获取 Pico 支付系统的支付结果和用户信息：

表 7.2 商户服务器需要实现的接口

名称	支付结果回调接口
请求类型	POST
请求 URL	支付，PayOrder 传入的参数 notify_url
请求格式	JSON



返回格式	JSON		
是否需要登录	是		
请求参数	详见“表 7.3 支付结果通知中的通知参数”		
请求参数示例			
返回参数	参数名	类型及范围	说明
	ret_code	string	错误代码。
	ret_msg	string	错误信息字符串。
	详见“表 7.4 返回结果”		
返回参数示例	<pre>{   "ret_code": "SUCCESS",   "ret_msg": "OK" }</pre>		
更新说明			

表 7.3 支付结果通知中的通知参数

字段名	变量名	必填	类型	描述
返回状态码	ret_code	是	String	SUCCESS/FAIL 此字段是通信标识, 非交易标识, 交易是否成功需要查看 result_code 来判断
返回信息	ret_msg	否	String	返回信息, 如非空, 为错误原因 签名失败参数格式校验错误
错误代码	sub_code	否	String	错误码
错误代码描述	sub_msg	否	String	错误返回的信息描述

字段名	变量名	必填	类型	描述
Pico 支付订单号	trade_no	是	String	Pico 支付订单号
商户订单号	out_trade_no	是	String	商户系统内部的订单号,
应用 ID	app_id	是	String	平台审核通过的应用 APP_ID
商户 ID	mch_id	是	String	支付分配的商户号
用户标识	open_id	是	String	用户在商户 appid 下的唯一标识
设备号	device_id	否	String	终端设备号
随机字符串	nonce_str	是	String	随机字符串, 不长于 32 位。推荐随机数生成算法
签名	signature	是	String	签名, 详见签名生成算法
业务结果	result_code	是	String	SUCCESS/FAIL
交易类型	trade_type	是	String	支付类型
货币类型	fee_type	是	String	货币类型
总金额	total_fee	是	String	订单总金额
实收金额	receipt_fee	是	String	实收金额
买家付款的金额	buyer_pay_fe	否	String	买家付款的金额

字段名	变量名	必填	类型	描述
	e			
代金券或立减优惠金额	coupon_fee	否	String	代金券或立减优惠金额
商家数据包	attach	否	String	商家数据包, 原样返回
支付完成时间	pay_time	是	String	支付完成时间, 格式为 yyyy-MM-dd HH:m m:ss

表 7.4 返回结果

字段名	变量名	必填	类型	描述
返回状态码	ret_code	是	String	SUCCESS/FAIL SUCCESS 表示商户接收通知成功并校验成功
返回信息	ret_msg	否	String	返回信息, 如非空, 为错误原因: 签名失败参数格式校验错误

特别提醒：商户系统对于支付结果通知的内容一定要做签名验证，防止数据泄漏导致出现“假通知”，造成资金损失。

签名校验规则是：

1. 返回的参数列表，去掉 signautre 参数，同时添加 key = "app\_secret" ,value=paykey  
，然后根据 key 值进行自然排序，多个参数之间用&隔开，最后进行 MD5 加密
2. 用加密后的字符串和获取到的 signature 进行比较

签名函数如下:

```
/**
 * result : 获取的数据的map集合
 * paykey : 就是开发者平台上的paykey
 */
public static String createSign(Map<String, Object> result, String paykey)
{
    if (result == null || result.size() == 0)
        return null;
    result.put("app_secret", paykey); //1. 添加key = "app_secret", value=payke
    String sign = result.get("signature"); //2. 保存signature的值, 用于校验
    result.remove("signature"); //3. 移除signature参数
    String[] tmp = new String[result.size()];
    int i = 0;
    for (String key : result.keySet())
    {
        tmp[i++] = key;
    }
    Arrays.sort(tmp); //4. 自然排序
    String sign = "";
    for (String string : tmp)
    {
        if (m.get(string) == null)
            continue;
        sign += string + "=" + URLEncoder.encode(m.get(string).toString()
            , "utf-8") + "&";
    }
    if (sign.endsWith("&"))
        sign = sign.substring(0, sign.length() - 1);
    Log.i(TAG, "createSign: " + sign);
    String localSign = MD5.MD5(sign); //5. 生成MD5加密后的字符串
    return localSign.equals(sign); //6. 和2中的sign进行校验
}
```

## 8 其他说明

### 8.1 Goblin HMD 按键



图 8.1 Goblin HMD 按键

表 8.1 GoblinHMD 按键映射

Goblin HMD 按键	说明
按键 1	Android 标准 POWER
按键 2	Android 标准 HOME

Goblin HMD 按键	说明
按键 3	
按键 4	Android 标准 VOLUME_UP
按键 5	Android 标准 VOLUME_DOWN

## 8.2 G2 HMD 按键

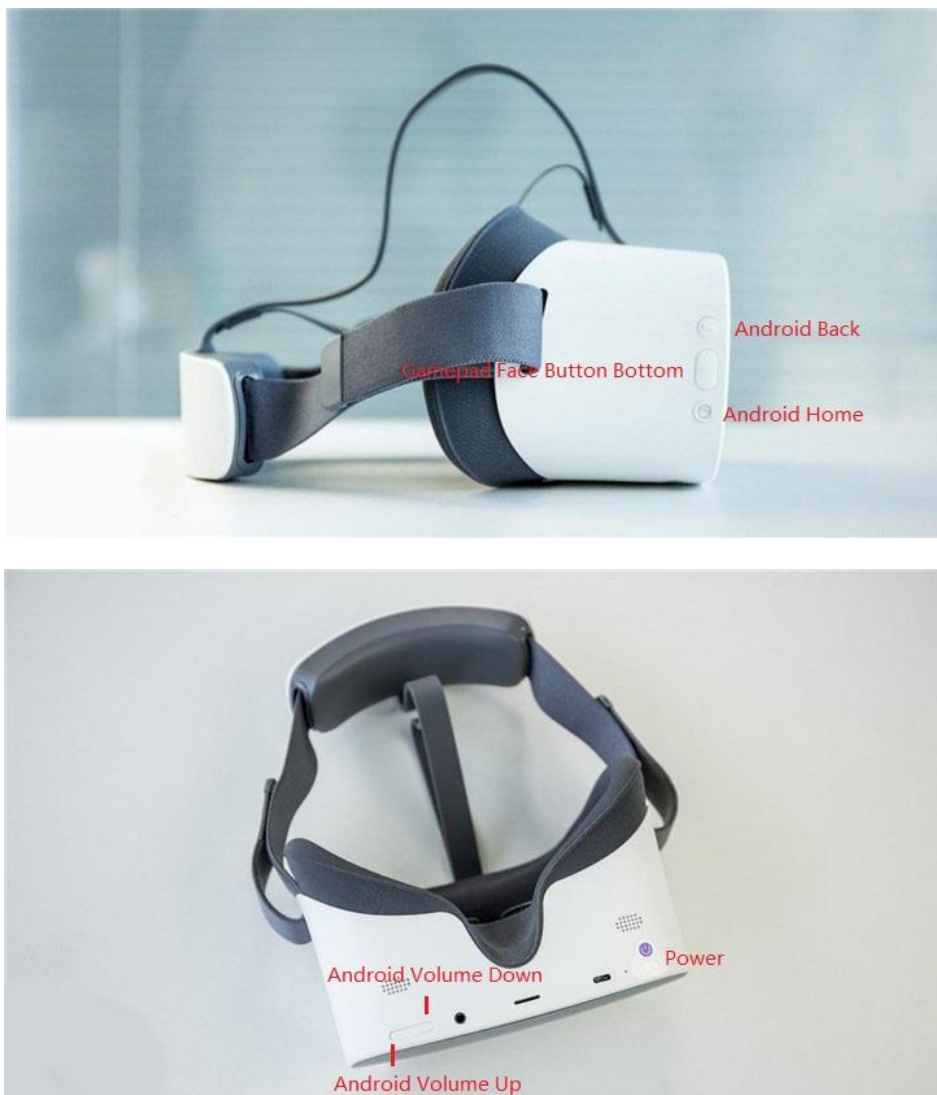


图 8.2 G2 HMD 按键

### 8.3 Pico Neo HMD 按键



图 8.3 Pico Neo HMD 按键

表 8.2 Pico Neo 按键映射

Goblin HMD 按键	说明
按键 1	Android 标准 VOLUME_UP
按键 2	Android 标准 VOLUME_DOWN
按键 3	Android 标准 BACK
按键 4	
按键 5	Android 标准 HOME

### 8.4 Pico Neo 安全区域

Pico Neo 安全区域类似 Oculus Rift Guardian System 与 SteamVR Chaperone System，可以在玩家走出一定的范围时给予提示，避免与房间墙壁等碰撞而发生危险。虽然 VR 一体机受内向外追踪的限制，并没有外置传感器传入实际的安全区域坐标，我们仍然建议开发者设置一个假定的安全范围，以保证用户的安全。

为此，我们提供了一个 Actor 蓝图类，用作安全区域，其实质上就是一个法线向内的圆筒，用来提醒用户安全预先定义的安全范围：

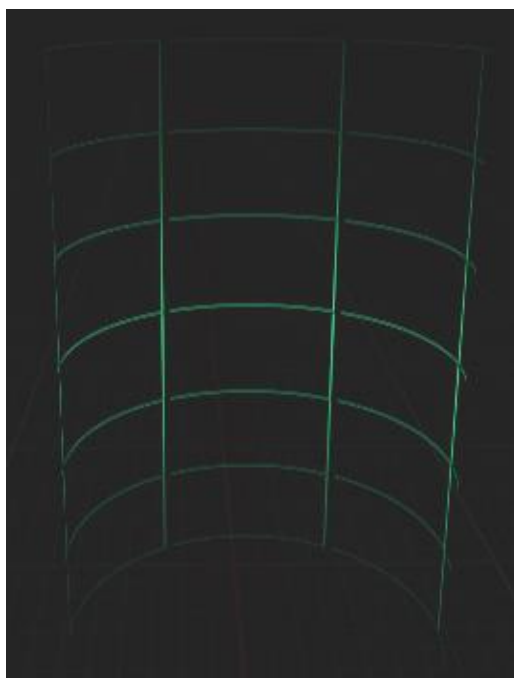


图 8.4 安全区域

建议在游戏关卡的 DefaultPawn 的 BeginPlay 事件后追加如下节点以添加安全区域：

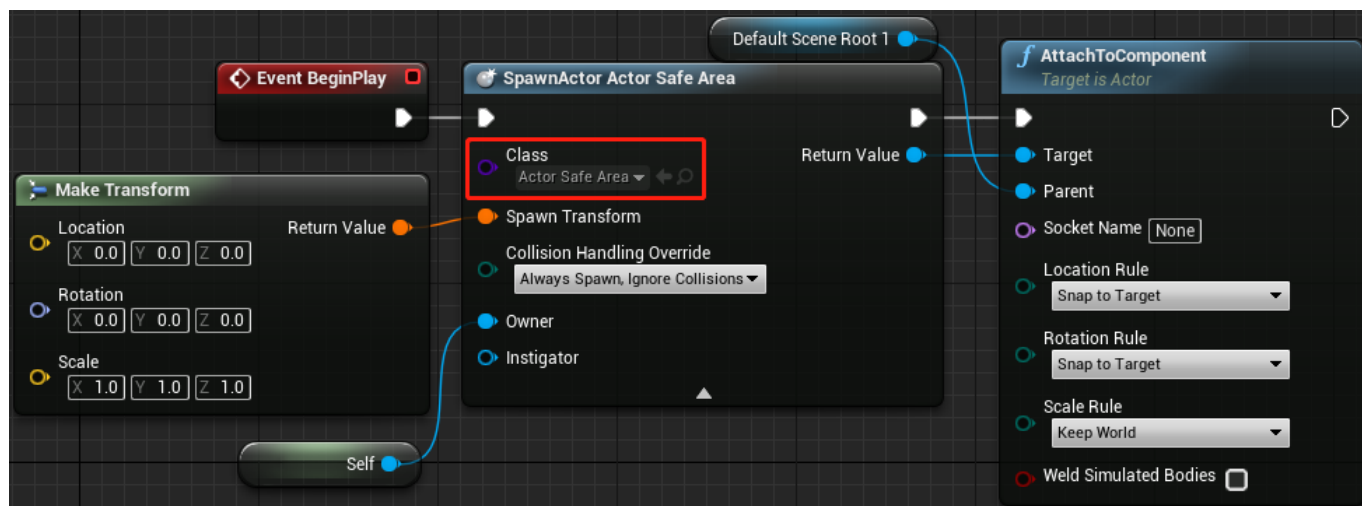


图 8.5 添加安全区域

然后，根据根据 Camera 或者 MotionController 与父组件的相对位置动态的改变安全区域的显示与隐藏。

另外，Actor\_SafeArea 蓝图类还提供了两个方法，用来获取及设置安全区域的半径：





图 8.6 安全区域相关函数

建议开发者将设置安全区域的功能开发给用户，让用户根据其所在环境空旷区域的实际大小设置游戏中安全区域的半径。

## 8.5 开启 Pico Neo 6 Dof 功能

SDK 默认的项目类型为头部 3Dof 跟踪、手部 3Dof 跟踪。对于具有 6Dof 的设备如 Pico Neo，如需开启 6Dof 功能，请执行如下操作：

打开菜单 Edit->Project Settings..., 找到 Plugins 子项，根据需要对 Pico Neo 下的选项进行勾选：

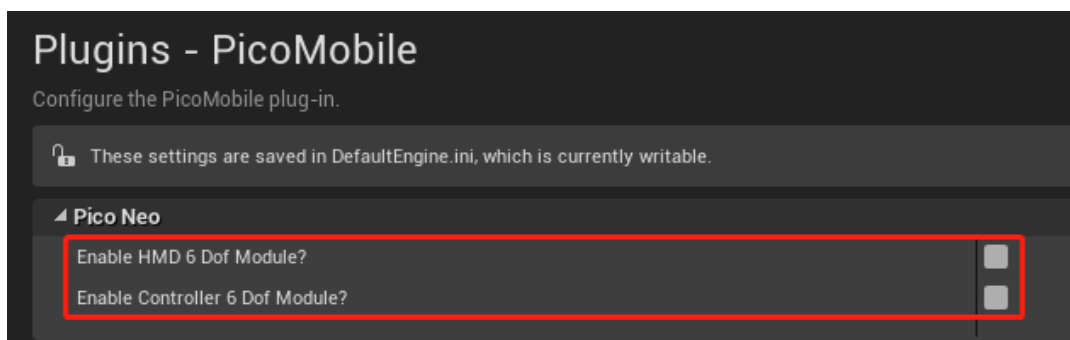


图 8.7 Pico Neo 6 Dof 选项

## 9 已知问题

- 不支持场景编辑器中的“虚拟现实预览”。

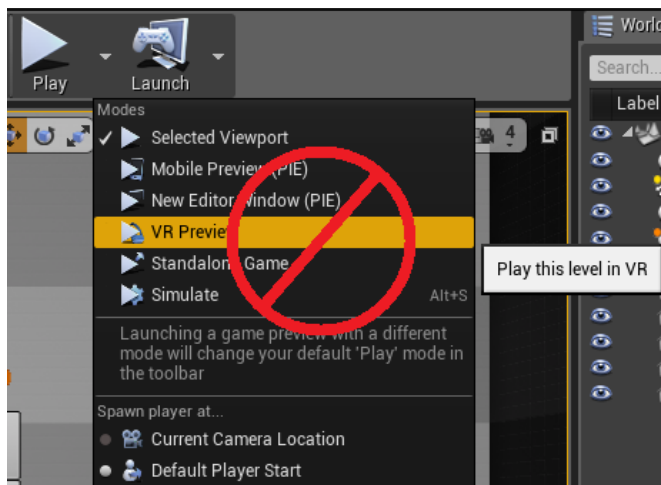


图 9.1 不支持虚拟现实预览

- 对于 UE4.18, 打包时请进入编辑>>项目设置>>平台>>Android, 取消勾选 Enable Gradle instead of Ant, 方能打包成功:



图 9.2 取消勾选 Enable Gradle instead of Ant

## 10 FAQ

**问：**为何项目打包后会出现 Android 系统的虚拟按钮？

**答：**勾选：项目设置→平台→Android→APKPackaging→Enable FullScreen Immersive on KitKat and above devices:

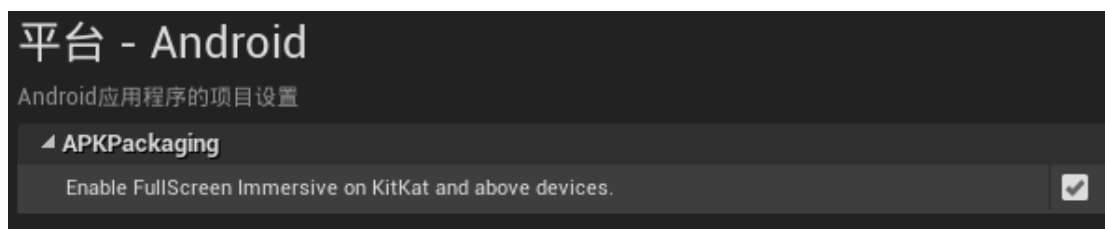


图 10.1 勾选 Enable FullScreen Immersive on KitKat and above devices

**问：**如何提高游戏的帧率？

**答：**提高游戏帧率可从以下两方面入手：

**1、关闭环境光遮蔽。**现阶段 VR 项目对全局光照的要求不是太高，可关闭环境光遮蔽，做法是打开项目设置，进入引擎/Rendering/Default Setting，取消勾选 Ambient Occlusion 与 Ambient Occlusion Static Fraction:

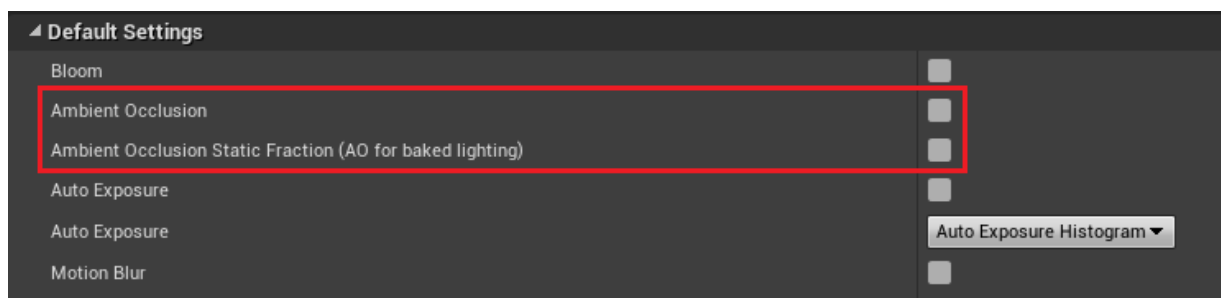


图 10.2 取消使用环境光遮蔽

**2、关闭 Mobile HDR。**做法是打开项目设置，进入引擎/Rendering/Mobile，取消勾选 Mobile HDR:

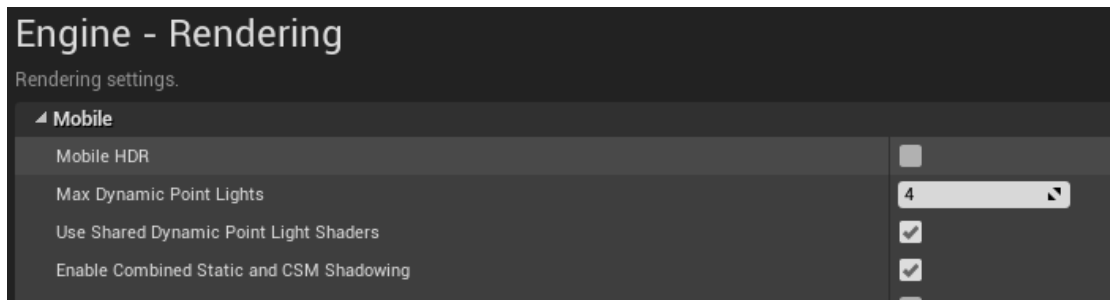


图 10.3 取消勾选 Mobile HDR

**问：**如何减小游戏包的大小？

**答：**减小游戏包的大小可从 3 个方面入手：

- 1、在项目设置/项目/打包中勾选 Create compressed cooked packages，压缩烘焙过的内容；
- 2、取消勾选未使用的 Plugins；
- 3、删除内容浏览器中没有用到的资源。

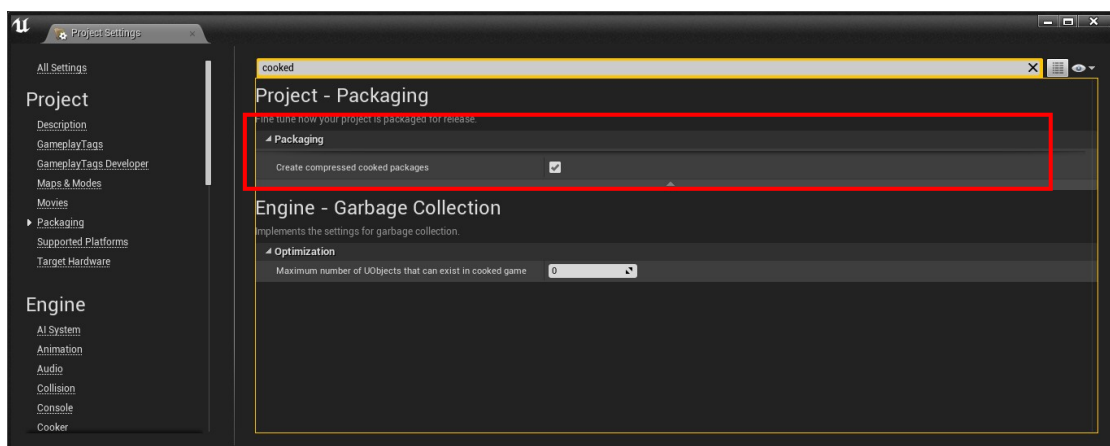


图 10.4 压缩烘焙过的内容

**问：**如何优化显示效果？

**答：**打开项目设置，进入引擎/Rendering/Mobile，调整 Mobile MSAA，可有效减少锯齿，提升显示效果。

倍数越高优化效果越好，但是会增加性能消耗，影响帧率。

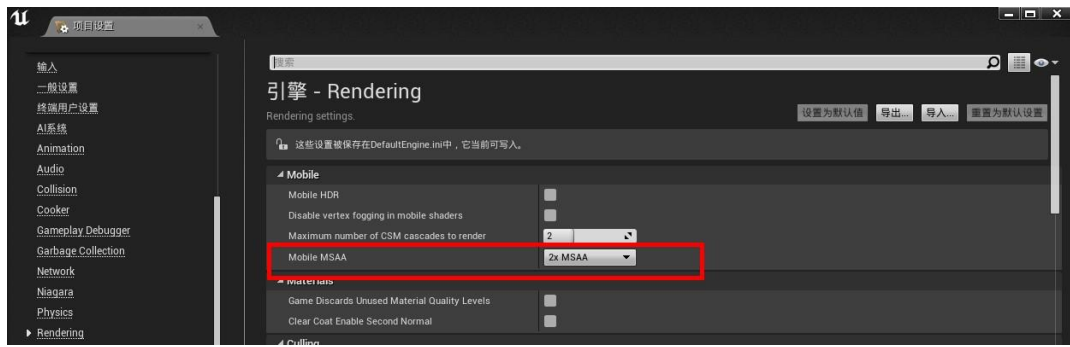


图 10.5 修改 Mobile MSAA

**问：**系统显示虚拟按键，怎么规避？

**答：**通过 UE 设置全屏模式，如下图。

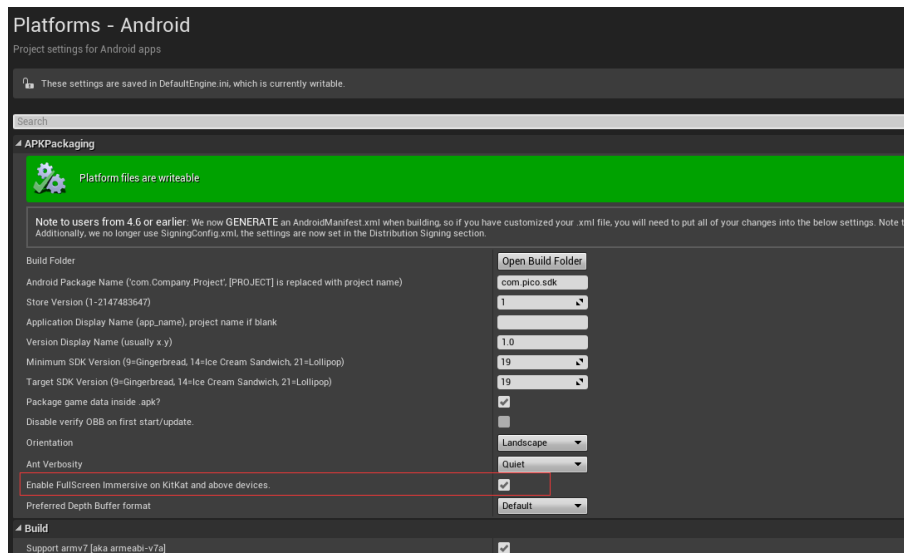


图 10.6 设置全屏模式

**问：**项目命名或路径中文，为什么使用 PicoVRSDK plugin 出现打包失败？

**答：**UE4 不支持中文，请不要使用中文和中文目录。

**问：**项目命名为 test，为什么使用 PicoVRSDK plugin 出现打包失败？

**答：**test 在 UE 中是命令关键字见下图，建议项目命名避开 UE 命令关键字。

```
// Configuration names:  
case "DEBUG":  
    Configuration = UnrealTargetConfiguration.Debug;  
    break;  
case "DEBUGGAME":  
    Configuration = UnrealTargetConfiguration.DebugGame;  
    break;  
case "DEVELOPMENT":  
    Configuration = UnrealTargetConfiguration.Development;  
    break;  
case "SHIPPING":  
    Configuration = UnrealTargetConfiguration.Shipping;  
    break;  
case "TEST":  
    Configuration = UnrealTargetConfiguration.Test;  
    break;
```

图 10.7 UE 命令关键字

**问:** 为什么游戏中 Goblin 手柄会时而闪烁一下?

**答:** 因为 UE4 游戏逻辑与渲染不在同一线程, Epic 为平滑运动控制器的移动, 默认会在渲染之前再更新一次运动控制器的位置和姿态, 但这种做法也产生了模型时而闪烁一下的 bug。要想规避这种闪烁, 只需选中 Motion Controller 组件, 在其细节面板中勾选 Disable Low Latency Update:

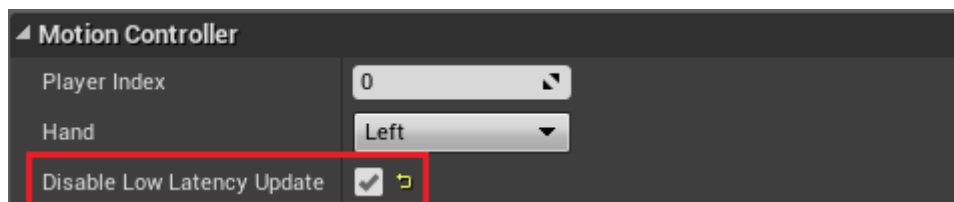


图 10.8 Disable Low Latency Update